



Guidelines

- ◆ We are a small group
- ◆ Interrupt the speaker
- ◆ Ask unrelated question
- ◆ Help us keep this fluid

- ◆ We can re-tool the agenda on-the-fly



Community Development



Topics

- ◆ Community Status
 - Why this is Important
 - Open Source Licensing
 - Contributors and Working Groups
 - Source Code
- ◆ Becoming a Developer
 - Graph and Rolls
 - Attributes
 - Command Line
- ◆ Avoiding becoming a Developer



Community Status



WHY?



Rocks is almost 10 years old!

- ◆ Mission accomplished
- ◆ Rocks is the *de facto* open-source clustering solution
- ◆ Great user community
 - ⇒ 2000+ on mailing list
 - ⇒ Amazing signal to noise ratio
- ◆ Everything from 2 nodes cluster to top 10 supercomputers



Rocks is almost 10 years old!

- ◆ 90% of development is
 - ⇒ NSF (and other grant) funded
 - ⇒ Located at UC San Diego
- ◆ Need to diversify development
 - ⇒ More ideas, passion, and focus areas
 - ⇒ More secure funding



OPEN SOURCE LICENSING



Licensing / Copyrights

http://www.rocksclusters.org/wordpress/?page_id=48

- ◆ Rocks is entirely open-source
- ◆ BSD Attribution License
 - ➔ Standard UNIX open-source
 - ➔ Very friendly for derived works
- ◆ We have not changed to the more recent non-attribution BSD license
- ◆ Copyrights are owned by University of California Regents
- ◆ 3rd party code is a mix of licenses and copyrights
 - ➔ Most of Rocks bits are 3rd party!



Attribution Clause

This product includes software developed by the Rocks® Cluster Group at the San Diego Supercomputer Center at the University of California, San Diego and its contributors.



Trademark

invent@ucsd.edu

- ◆ The Rocks name and logo are registered trademarks.
- ◆ For fee licensing is available
 - Standard usage
 - Derivative usage (e.g. “ACME Rocks”)



Summary

- ◆ Rocks is open-source and free
- ◆ Use it any way you wish
- ◆ Make billions of dollars with it without even buying us a single beer
- ◆ Give **us attribution**
- ◆ License the name for commercial use
- ◆ These two things help keep us funded



EARLY COMMUNITY ROLLS



Sun Grid Engine

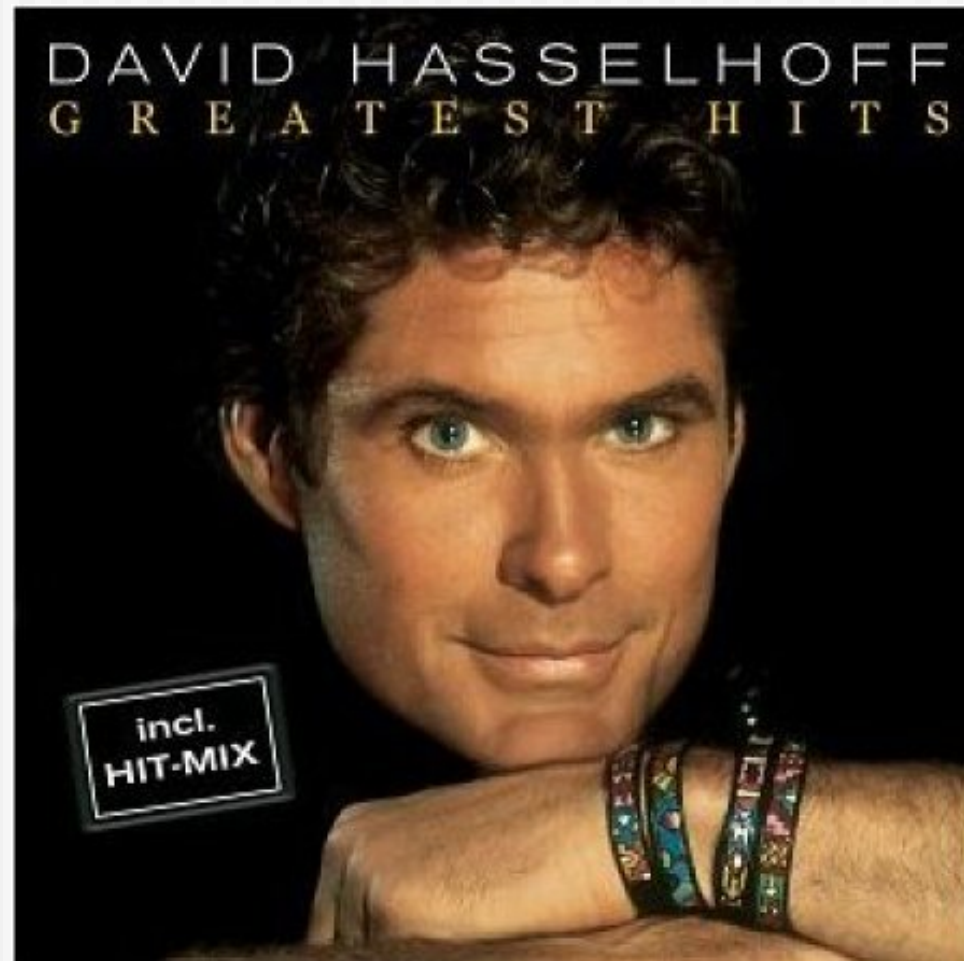
- ◆ Way back in 2004
- ◆ Rocks supported PBS
- ◆ Scalable Systems added SGE support
 - Laurence Liew, Najib Ninaba
 - 1st external developers for Rocks
 - Based in Singapore
- ◆ SGE Roll created from this



Torque Roll

- ◆ Way back in 2006
- ◆ SGE Roll was the favorite of core team
- ◆ Threatened to drop PBS Support
- ◆ The Computer Center, University of Tromsø
 - ⇒ Roy Dragseth
 - ⇒ 2nd major external developer
 - ⇒ Based in Norway
- ◆ Continues to develop and support Torque Roll

We were “big” overseas





First major attempt for actively recruit developers

WORKING GROUPS



Working Groups

◆ Purpose

- ⇒ Fill gaps from core development team
- ⇒ Handle issues off the core road map
- ⇒ Make Rocks a more flexible solution

◆ Success Metrics

- ⇒ Number of Rolls produced
- ⇒ Amount of new Documentation (Wiki, ...)



Software Update WG

[https://wiki.rocksclusters.org/wiki/index.php/Software_Update_\(SUWG\)](https://wiki.rocksclusters.org/wiki/index.php/Software_Update_(SUWG))

- ◆ Started early 2008
 - ⇒ Threads on yum updates increased
 - ⇒ Core team said “don’t do it”
 - ⇒ Advocates said “it works for me”
 - ⇒ WG was recruited to address the issue
- ◆ Best practices defined
 - ⇒ Exclude lists
 - ⇒ Additional docs on custom restore Rolls



Status

- ◆ Community Interest: High
- ◆ Documentation: Moderate
- ◆ Rolls Produced: None

- ◆ Summary: Some real interest but needs leadership.



Thumper Working Group

https://wiki.rocksclusters.org/wiki/index.php/Rocks_on_Thumper

- ◆ Began with Rocks Solaris port
 - ⇒ Sun funded
 - ⇒ How to Manage ZFS NAS appliances
 - ⇒ Specifically Sun Thumper
- ◆ Core team lead effort
- ◆ Used by several groups at UCSD
- ◆ Software is released



Status

- ◆ Community Interest: Low
- ◆ Documentation: Good
- ◆ Rolls Produced: Jumpstart
- ◆ Summary: Excellent activity with a small UCSD audience. Needs to build a larger user base.



Rolls Working Group

https://wiki.rockclusters.org/wiki/index.php/Rolls_Working_Group

- ◆ Started early 2009
 - ⇒ Developing free versions of commercial Rolls
 - ⇒ Organized by Stanford University
- ◆ Self-organized group of a 3 individuals
- ◆ Good initial offering of Rolls
- ◆ Struggled with mailing list support



Status

- ◆ Community Interest: High
- ◆ Documentation: Average
- ◆ Rolls Produced: Good
- ◆ Summary: Excellent start, needs help with user support and keeping current with Rocks releases.



Triton Working Group

<http://tritonresource.sdsc.edu/>

- ◆ Started 2009
 - ⇒ Developing Roll for large production cluster
 - ⇒ Every piece of SW on system is part of a Roll
 - ⇒ Includes commercial software
- ◆ Amazing set of Rolls (20+) to be released
- ◆ Triton group is here at SDSC
- ◆ No organized presence on Rocks list



Status

- ◆ Community Interest: Good
- ◆ Documentation: Good
- ◆ Rolls Produced: Excellent
- ◆ Summary: Highly productive group, but meets weekly with member(s) of Rocks core team. Phil is also their boss.



Great idea, some good traction, but not what we want

OVERALL GRADE: C-



What can UCSD do better?

- ◆ WG phone/video conference
 - WG to Core team
 - WG All hands
- ◆ Need to communicate roadmaps between WGs and Core team
 - Ease release tracking
 - No surprises (e.g. Rocks Command Line)
- ◆ Where should support issues go?
 - Main list
 - A new WG list



Ideas?

- ◆ Docs in dev process
 - ⇒ Devel guide out of date
 - ⇒ Mine mailing list for solution
- ◆ Developer Cloning Process
 - ⇒ Jumpstart guide to development
- ◆ RESOLVED tag on mailing list
- ◆ Bug/Issue searchable database
 - ⇒ RH is a good example of this

◆ IRC

◆ AIM Address Book (non-indexed)



What can you do better

- ◆ Tell us what you want
 - ⇒ Complain
 - ⇒ A lot
 - ⇒ But, nicely
- ◆ Ask for help to start a new working group
- ◆ Join an existing working group
- ◆ We are starting this **today**



SOURCE CODE

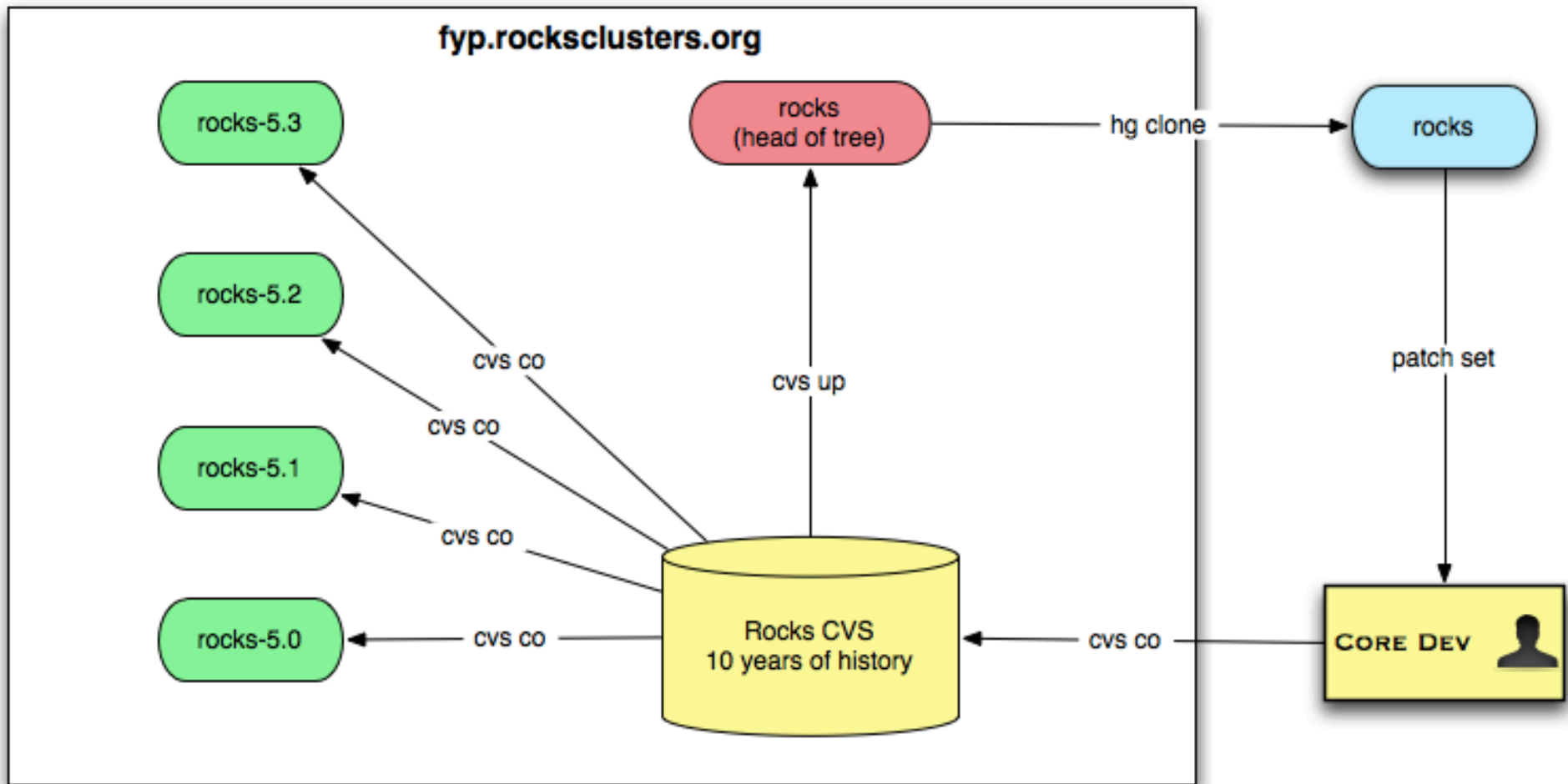


Version Control

- ◆ Core team uses CVS
 - ⇒ That's what we started with
 - ⇒ We aren't changing anytime soon
- ◆ CVS access available to very few people
 - ⇒ Too risky
 - ⇒ Access control is a pain
 - ⇒ Release management difficult
- ◆ We use Mercurial for all non-core development
- ◆ Mercurial synced to CVS every 10 minutes!



Workflow





Example

```
$ hg clone http://fyp.rocksclusters.org/hg/rocks-5.3
destination directory: rocks-5.3
real URL is http://fyp.rocksclusters.org/hg/
rocks-5.3/
requesting all changes
adding changesets
adding manifests
adding file changes
added 1 changesets with 2815 changes to 2815 files
2815 files updated, 0 files merged, 0 files removed,
0 files unresolved
```




Issues

- ◆ Mecurial is slow
- ◆ Transaction based
 - ⇒ Any aborted operation rolls back
 - ⇒ Do not stop the clone
- ◆ Patch sets can be tedious



Advantages - Freedom

- ◆ Publish your own repository
- ◆ No need to even commit back to core
- ◆ Commit broke code and only hurt yourself
- ◆ Core Rocks remains stable
- ◆ HG clones innovate



Notes

- ◆ For code older than 5.0
 - ⇒ <ftp://ftp.rocksclusters.org/pub/rocks/rocks-src>
 - ⇒ rocks-2.3 to rocks-4.3
- ◆ We are not tied to this workflow
- ◆ We are not tied to HG
- ◆ Other workflow suggestions are solicited



Becoming a Developer

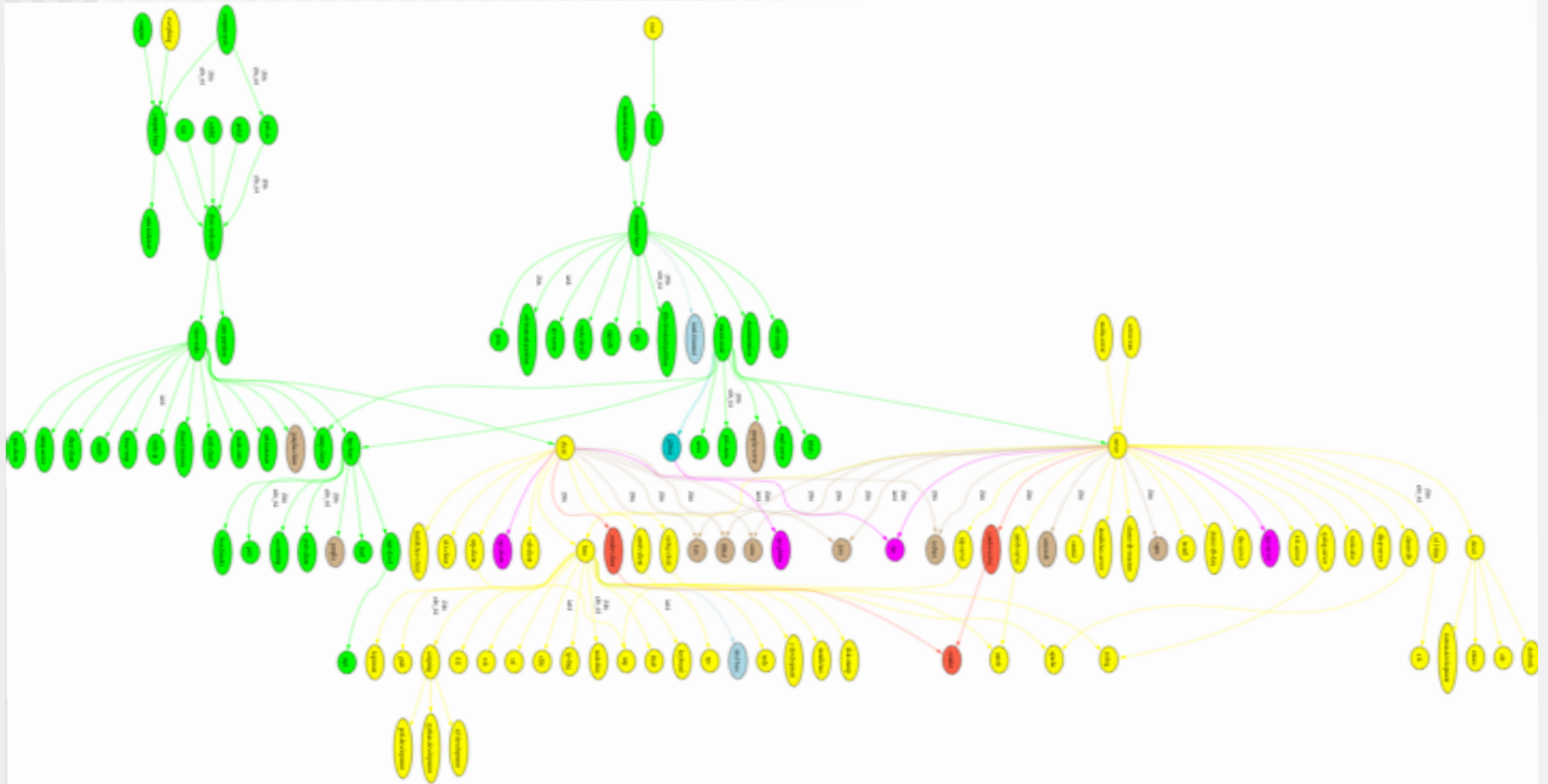
Resume @ 11:20



The Rocks engine

GRAPH AND ROLLS

Rocks Configuration Graph





The XML Graph Includes

◆ Nodes

- Single purpose modules
- Kickstart file snippets (XML tags map to kickstart commands)
- Approximately 200 node files in Rocks

◆ Graph

- Defines interconnections for nodes
- Think OOP or dependencies (class, #include)
- A single default graph file in Rocks

◆ Macros

- SQL Database holds site and node specific state
- Node files may contain `&state;` entities (attributes)

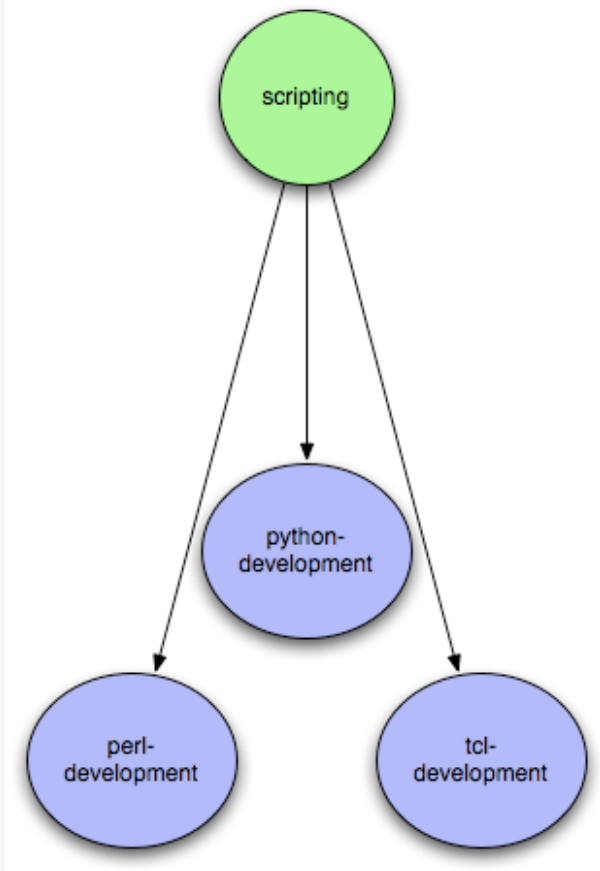


Composition

◆ Aggregate Functionality

◆ scripting IsA

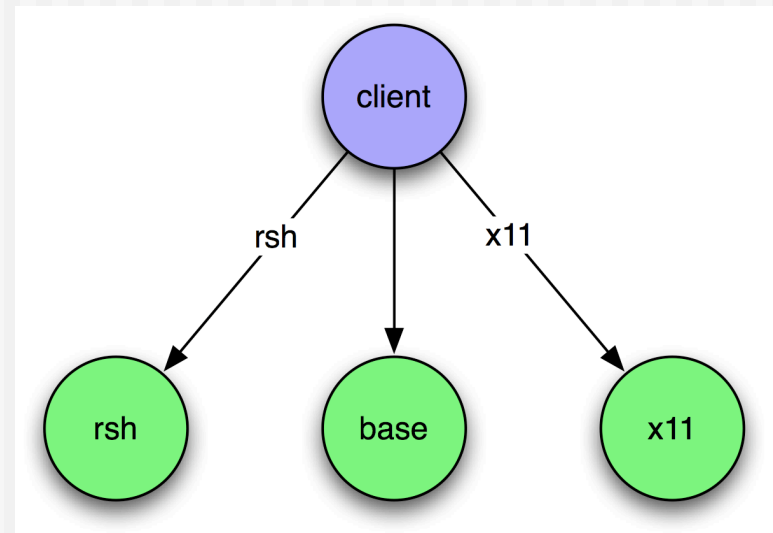
- perl-development
- python-development
- tcl-development





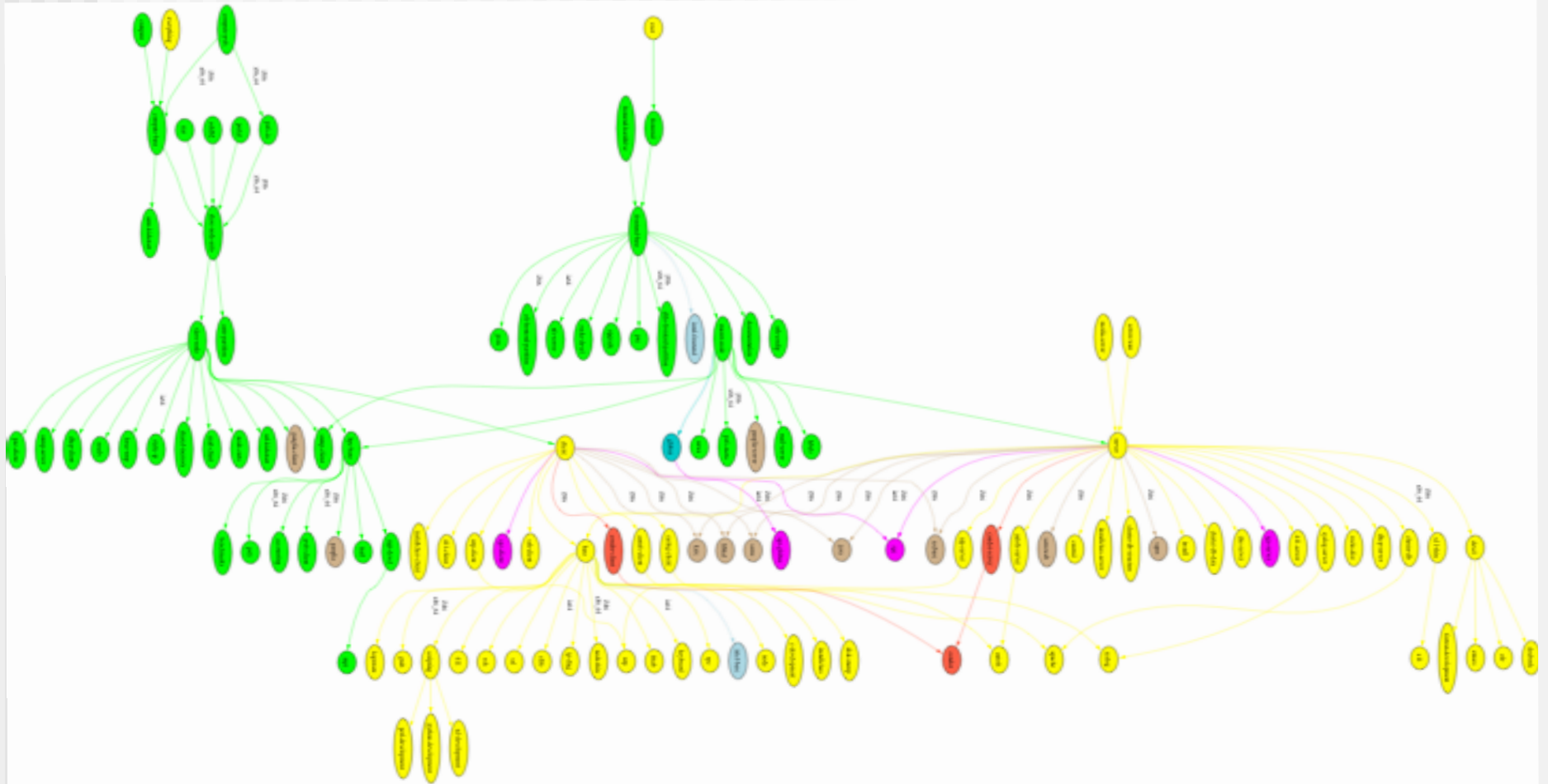
Traverse by Attributes

- ◆ if `x11 == TRUE`
 - `client IsA x11`
- ◆ if `rsh == FALSE`
 - `client IsNotA rsh`
- ◆ **Most important slide in this session**
- ◆ RCL allows you to control the graph





Think of this as Cluster DNA

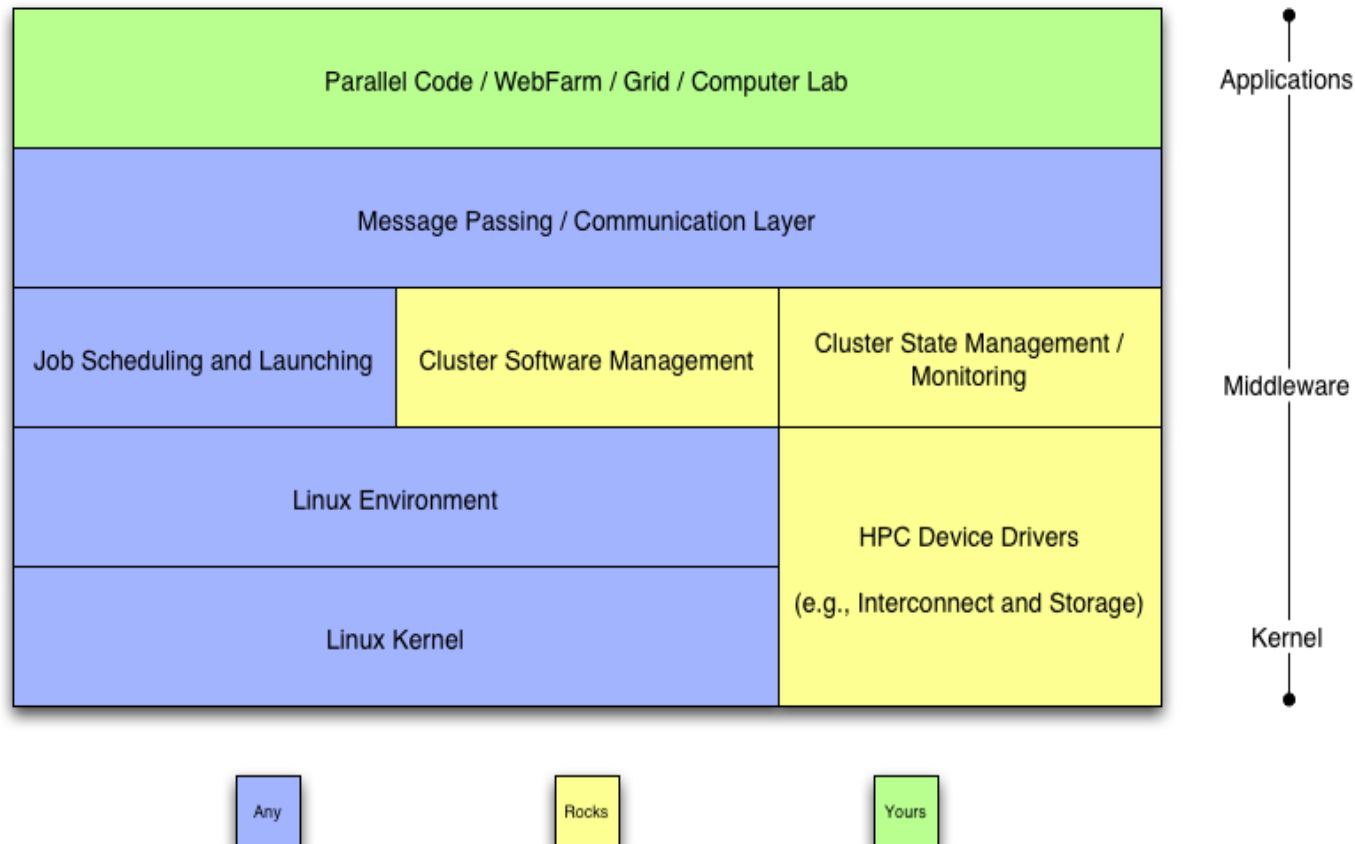




ROLL FUNDAMENTALS

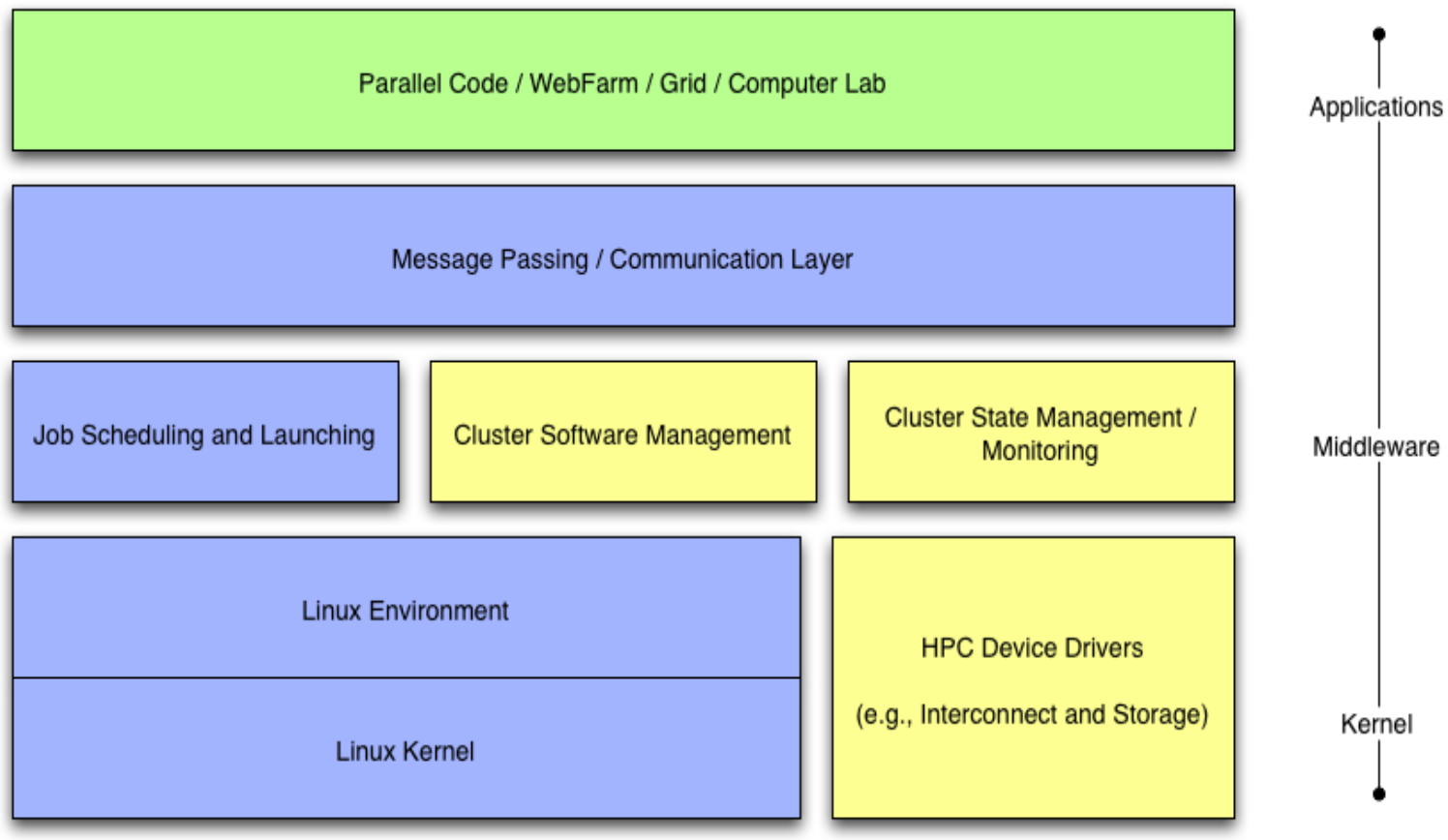


Cluster Software Stack





Rolls Break Apart Rocks



Rolls: Modifying a Standard System Installer to Support User-Customizable Cluster Frontend Appliances. Greg Bruno, Mason J. Katz, Federico D. Sacerdoti, and Phil M. Papadopoulos. *IEEE International Conference on Cluster Computing*, San Diego, California, Sep. 2004.



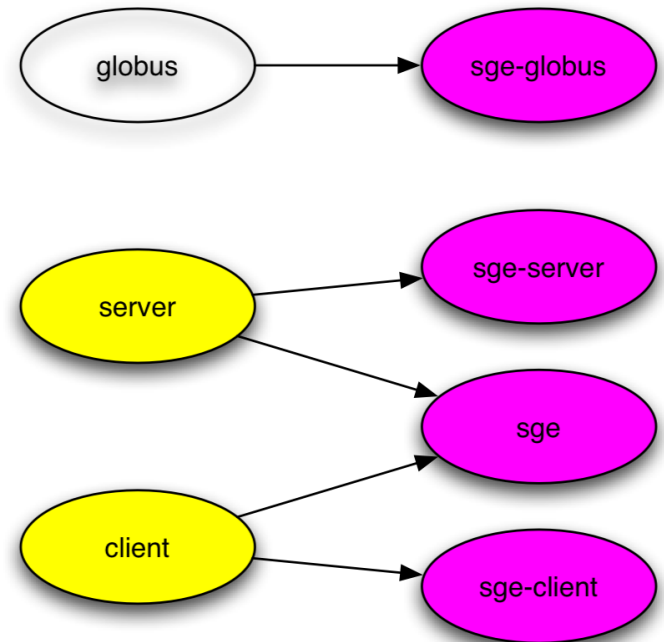
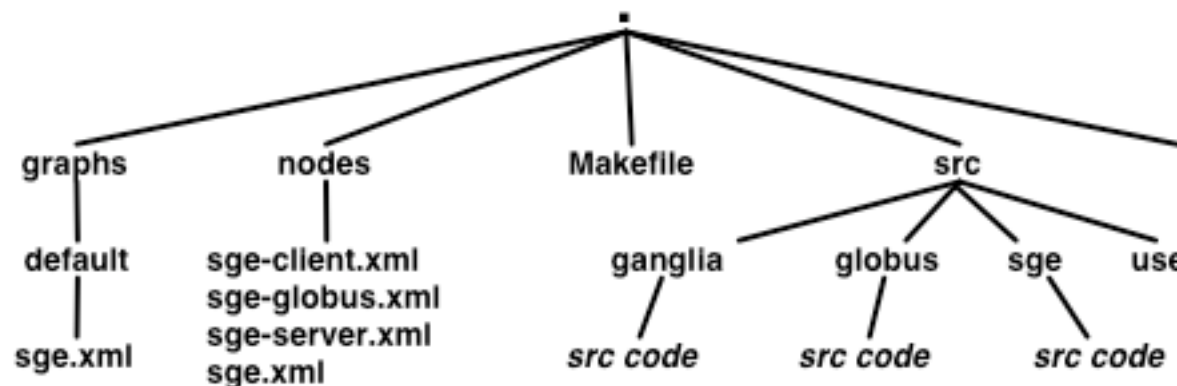
Our Graph Has Colors

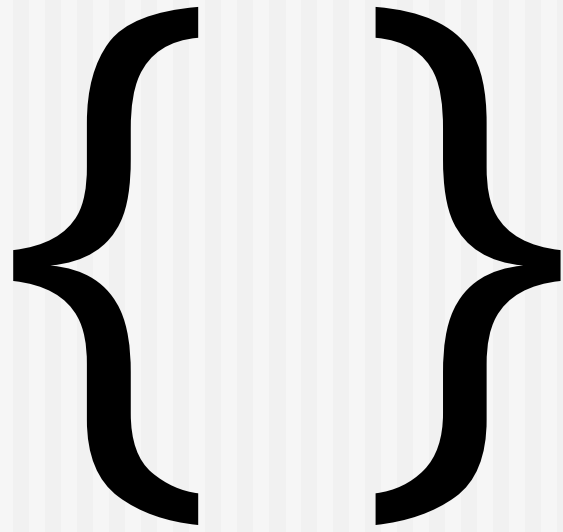




Rolls are sub-graphs

- ◆ A graph makes it easy to 'splice' in new nodes
- ◆ Each Roll contains its own nodes and splices them into the system graph file

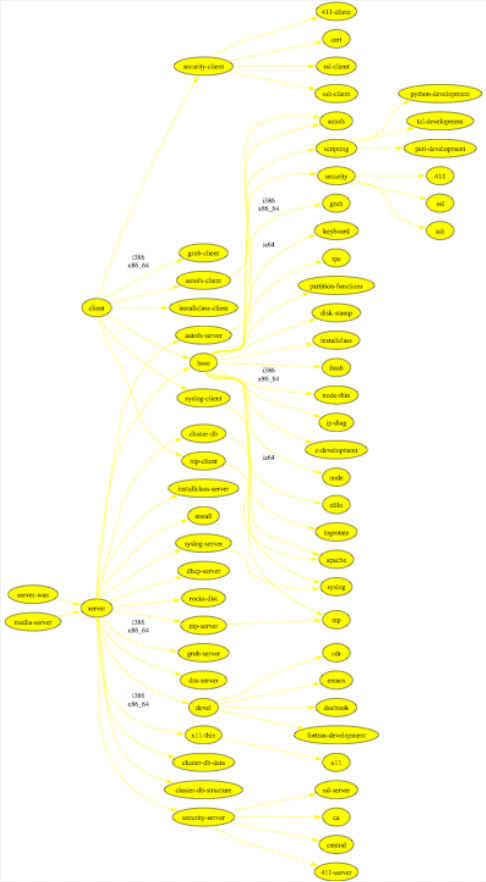




STARTING FROM THE EMPTY SET



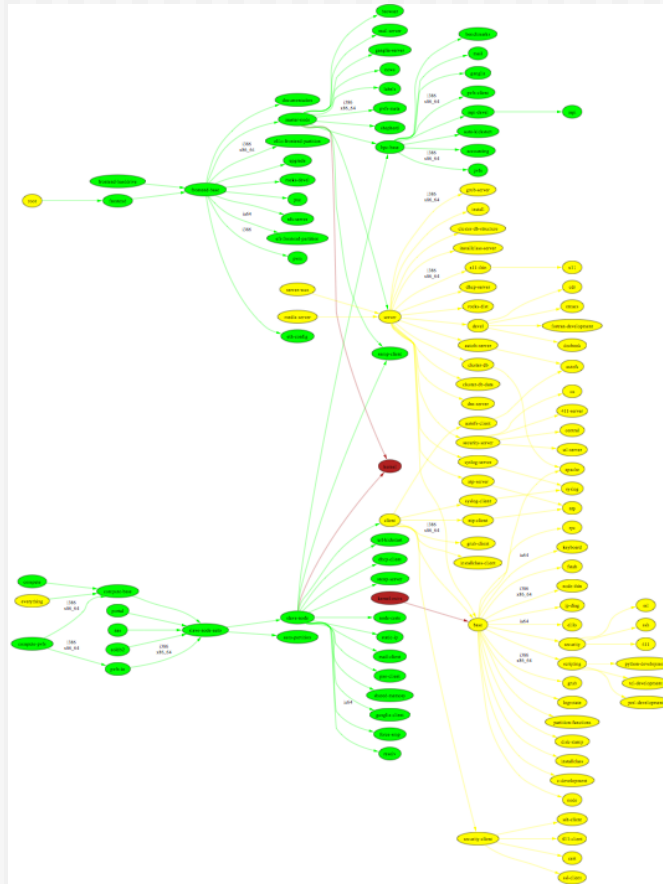
{ base }





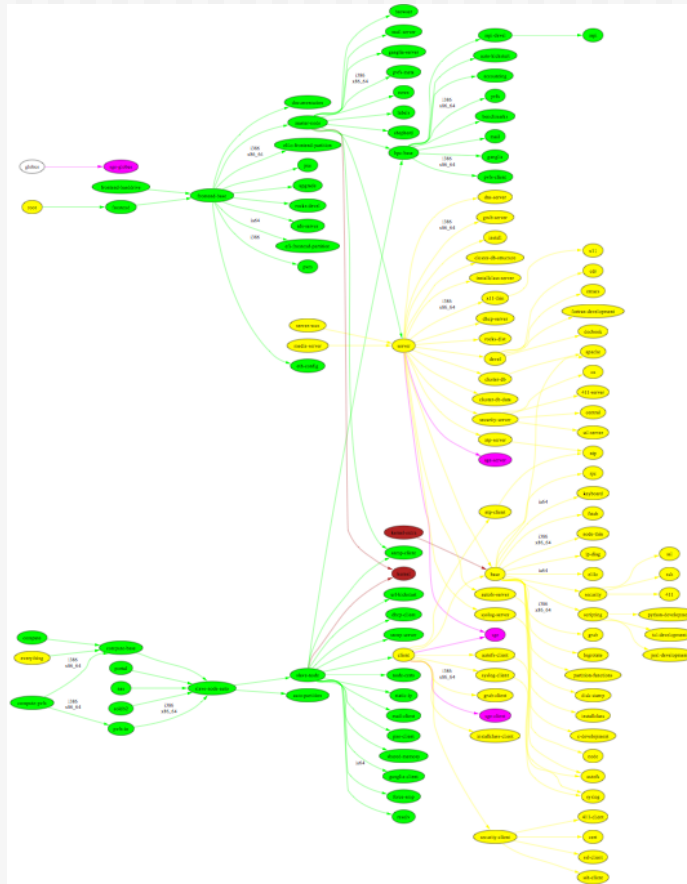


{ base, hpc, kernel }





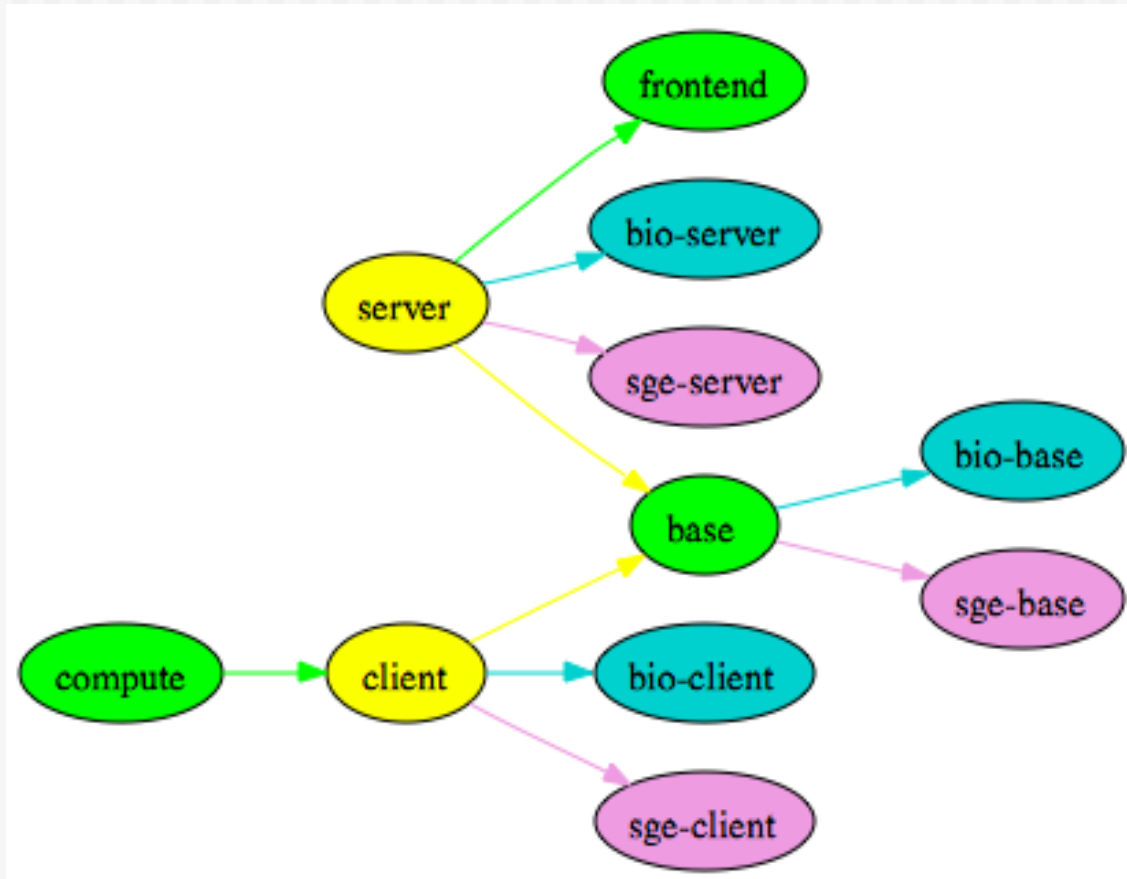
{ base, hpc, kernel, sge }





Simplified Example

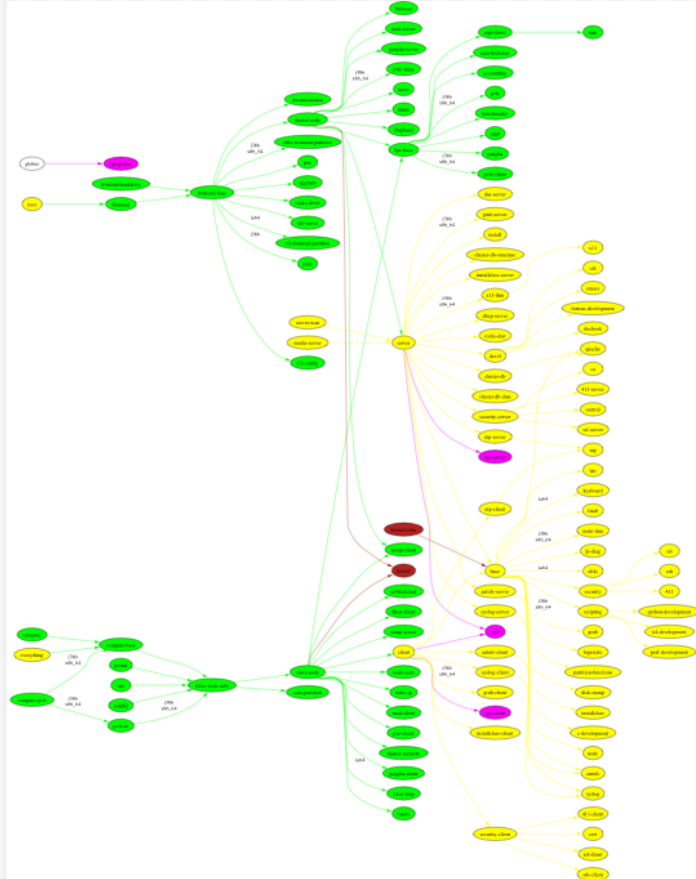
{base, hpc, sge, bio}



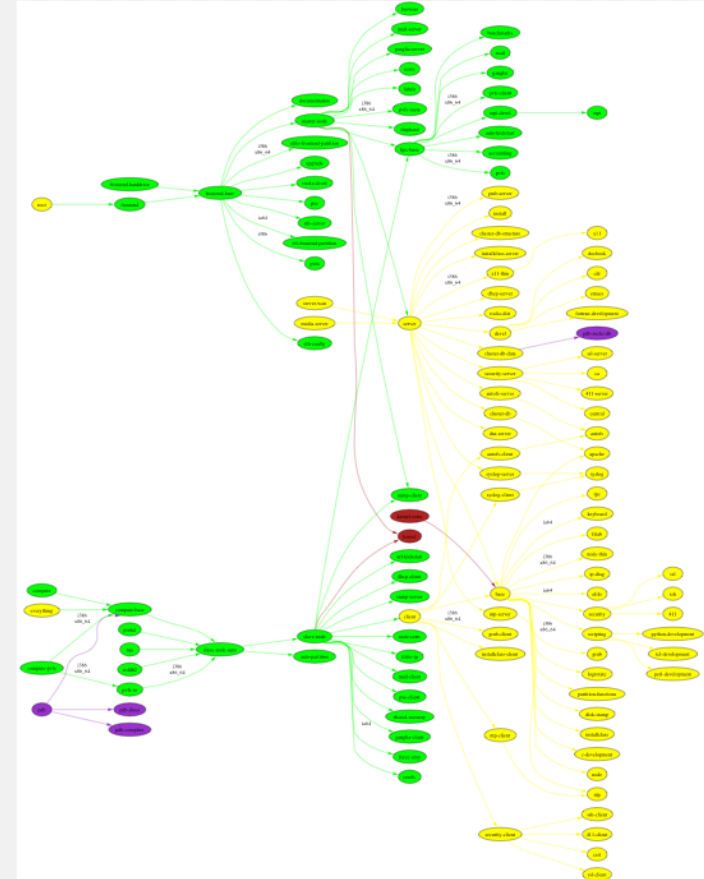


Two different Clusters

MPI Cluster:::{base, hpc, kernel, sge}



Protein Databank:::{base, hpc, kernel, pdb}





ATTRIBUTES



Attributes

◆ Attributes can be set at 4 levels:

- ⇒ Globally
 - 'rocks set attr'
- ⇒ By appliance type
 - 'rocks set appliance attr'
- ⇒ By OS (linux or sunos)
 - 'rocks set os attr'
- ⇒ By host
 - 'rocks set host attr'



Attributes

- ◆ Example, set the public IP address of a remote frontend that is used during a 'central' installation:

```
# rocks set host attr vi-1.rockclusters.org \  
  Kickstart_PublicAddress 137.110.119.118
```



Attributes

```
# rocks list host attr tile-0-0
HOST      ATTR                                VALUE                                SOURCE
tile-0-0: Info_CertificateCountry          US                                  G
tile-0-0: Info_CertificateLocality          San Diego                          G
tile-0-0: Info_CertificateOrganization      CalIT2                             G
tile-0-0: Kickstart_DistroDir               /export/rocks                      G
tile-0-0: Kickstart_PrivateAddress          10.1.1.1                           G
tile-0-0: Kickstart_PrivateBroadcast        10.1.255.255                       G
tile-0-0: Kickstart_PrivateDNSDomain        local                               G
tile-0-0: Kickstart_PrivateDNSServers       10.1.1.1                           G
tile-0-0: Kickstart_PrivateGateway          10.1.1.1                           G
tile-0-0: Kickstart_PublicDNSServers        132.239.0.252                      G
tile-0-0: Kickstart_PublicGateway           137.110.119.1                      G
tile-0-0: Kickstart_PublicHostname           vizagra.rocksclusters.org          G
tile-0-0: Kickstart_PublicKickstartHost     central.rocksclusters.org          G
tile-0-0: Kickstart_PublicNTPHost           pool.ntp.org                       G
tile-0-0: Kickstart_PublicNetmask           255.255.255.0                     G
tile-0-0: Kickstart_PublicNetmaskCIDR       24                                  G
tile-0-0: Kickstart_PublicNetwork           137.110.119.0                     G
tile-0-0: Kickstart_Timezone                America/Los_Angeles                G
tile-0-0: Server_Partitioning               force-default-root-disk-only       G
tile-0-0: arch                             x86_64                             H
tile-0-0: hostname                         tile-0-0                           I
tile-0-0: rack                             0                                  I
tile-0-0: rank                             0                                  I
tile-0-0: rocks_version                     5.2                                G
tile-0-0: HideBezels                       false                               G
tile-0-0: HttpConf                         /etc/httpd/conf                    O
tile-0-0: HttpConfigDirExt                  /etc/httpd/conf.d                  O
tile-0-0: HttpRoot                          /var/www/html                      O
```



Edge Conditionals

- ◆ Use attributes to conditionally traverse edges of the configuration graph

```
<edge from="client" cond="rsh">  
  <to>rsh</to>  
</edge>
```

- ◆ If 'rsh' evaluates to 'true', then the edge from 'client' to 'rsh' will be traversed
 - ➡ Default value is 'false'



Edge Conditionals

- ◆ To set a conditional attribute:

```
# rocks set attr rsh true
```

- ◆ Edge conditionals are attributes
- ◆ Can also be set at 4 levels:
 - ⇒ Globally
 - ⇒ By appliance type
 - ⇒ By OS (linux or sunos)
 - ⇒ By host



COMMAND LINE



Evil Commands

```
Usage: add-extra-nic [-hvv] [-p password] [-u host] [-d database] [--help]
[--list-rcfiles] [--list-project-info] [--verbose] [--dump] [--del] [--list]
[--verbose] [--no-update] [--no-modify] [--dryrun] [--rcfile arg] [--host host]
[--password password] [--db database] [--user host]
[--if interface (default: eth1)] [--mac mac address]
[--module linux driver module name] [--ip ip address]
[--netmask netmask (default /24)] [--gateway ip address of gateway]
[--name hostname on new interface] [--site client ip] node
```

```
Usage: rocks-dist [-hvcvpv] [-p password] [-u host] [-d database] [-a arch]
[-d dirname] [-g path] [-l lang] [-r release] [--help] [--list-rcfiles]
[--list-project-info] [--verbose] [--copy] [--debug] [--graph-draw-invis-edges]
[--graph-draw-order] [--graph-draw-edges] [--graph-draw-key] [--graph-draw-all]
[--graph-draw-landscape] [--install] [--verbose] [--with-rolls-only] [--clean]
[--notorrent] [--rcfile arg] [--host host] [--password password]
[--db database] [--user host] [--arch architecture] [--comps path]
[--dist dirname] [--graph-draw-size arg] [--graph-draw-format arg]
[--mirror-dir dirname] [--mirror-host hostname] [--root dirname]
[--cdrom /mnt/cdrom] [--with-roll rollname-rollversion]
[--path single path item] command
```

Available commands:

dist dvd makecontrib makesitenodes copycd usb copyroll cdrom paths graph dist2mirror



Command Line as API

- ◆ Lack of consistency in Rocks commands
 - ⇒ add-extra-nic (15 flags)
 - ⇒ 411put
 - ⇒ rocks-dist
 - ⇒ dbreport (~ a dozen reports)
- ◆ Not extensible to other groups
 - ⇒ How do I add a flag to an existing command?
 - ⇒ How do I add a new command?
 - ⇒ How do I document my command?



Do Over

- ◆ Consistent
 - Interface
 - Argument parsing
 - Usage / Help
- ◆ Extensible
 - Easy to add commands (3rd party rolls)
 - Easy to modify commands
- ◆ Easy to guess the right command
- ◆ Purge all –flags from Rocks
- ◆ Hide the SQL database (and underlying schema)
- ◆ Inspired by Trac



Verb Based

- ◆ “add”, “set”, “enable”, ...
 - ⇒ Modify the cluster database
- ◆ “list”, “dump”, “report”
 - ⇒ Inspect the cluster database
- ◆ About 20 verbs in the command line so far
- ◆ You can even add your own





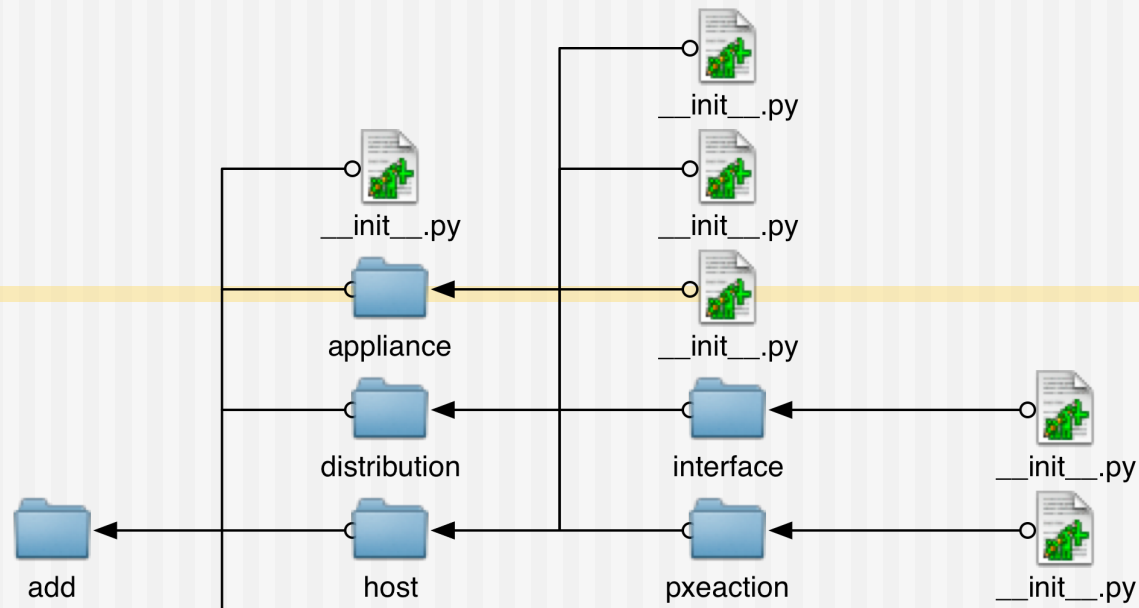
Grammar

- ◆ rocks <verb> <object...> <subject> <params...>
- ◆ Object is general to specific
 - ⇒ “host” “interface”
 - ⇒ “network” “subnet”
 - ⇒ “viz” “layout”
- ◆ Subject is typed
 - ⇒ host
 - ⇒ appliance
 - ⇒ network

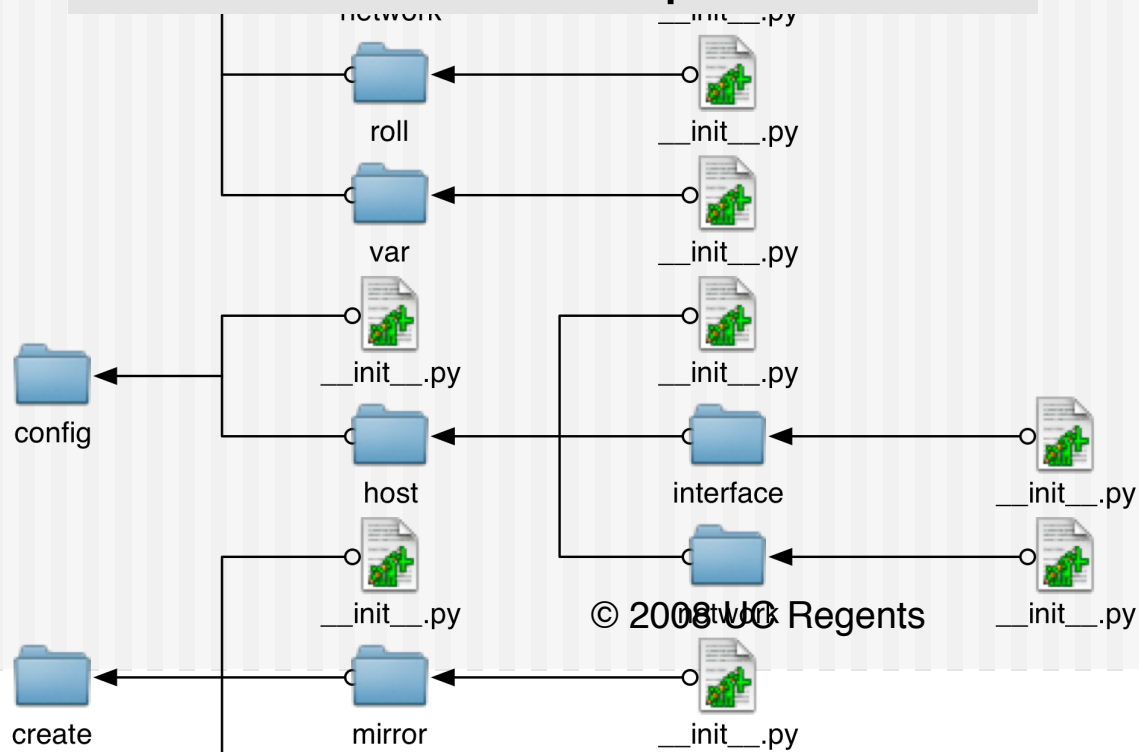


Implementation

- ◆ Python
 - ⇒ Similar to existing dbreport code
 - ⇒ Very small modules
- ◆ Command line is identical to the directory hierarchy
 - ⇒ Verbs are directories
 - ⇒ Objects are directories
 - ⇒ Subjects are `__init__.py` files
- ◆ Commands are added by adding directories



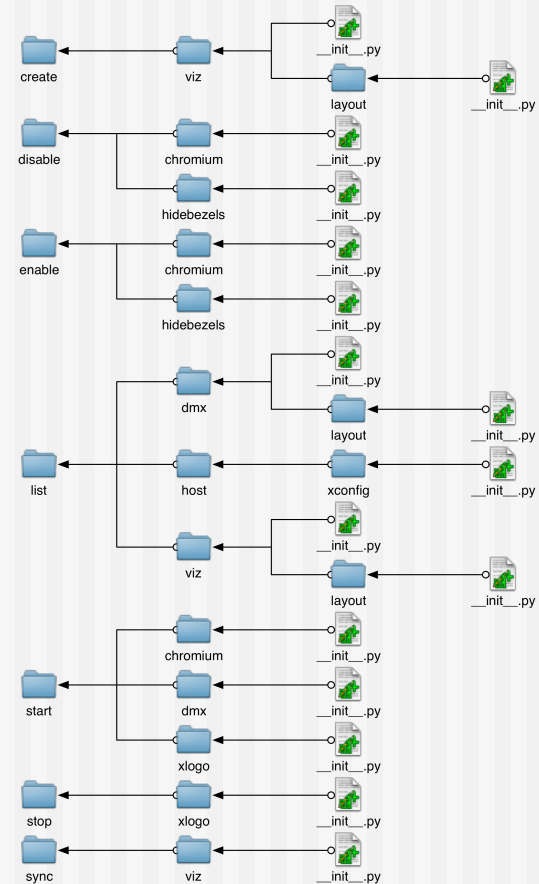
rocks add host pxeaction





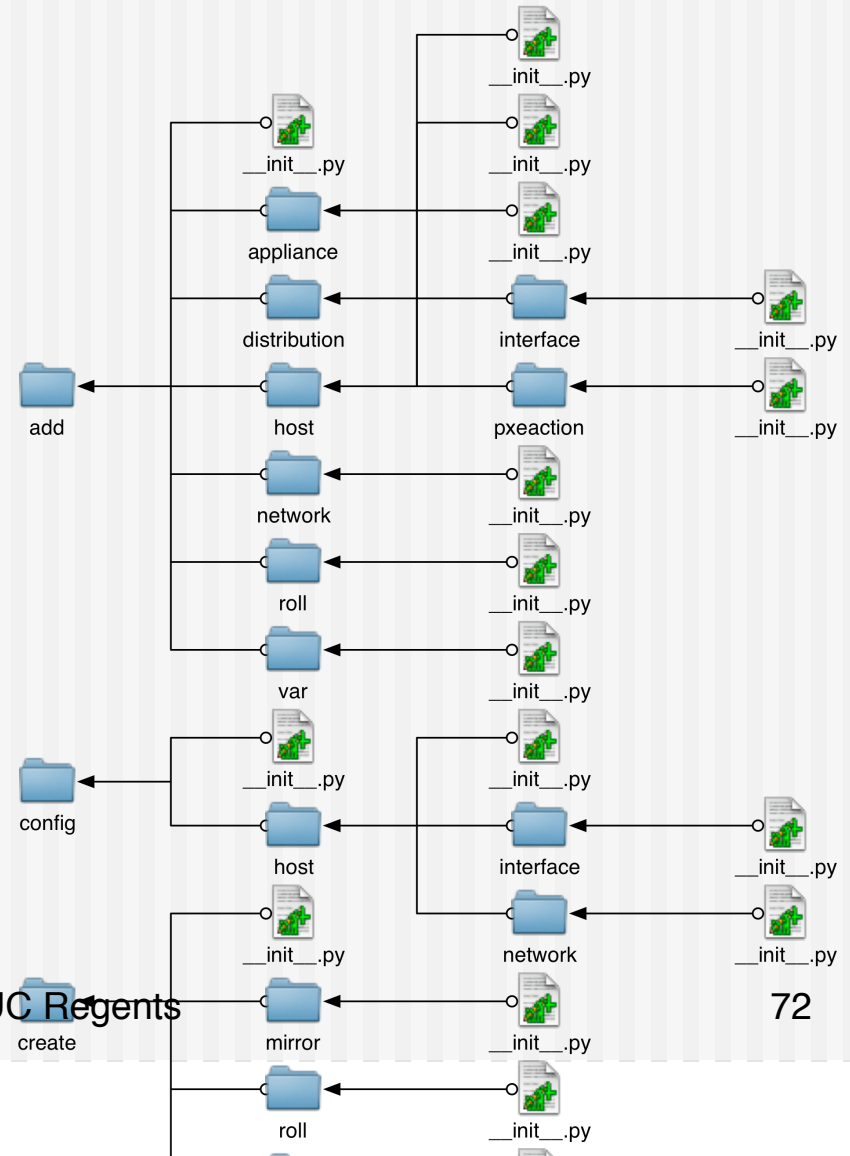
Rolls Can Add Commands

- ◆ Similar to the configuration graph
- ◆ Rolls can add command line
 - ➔ Files : commands
 - ➔ Directories : verbs and objects
- ◆ Think hard before adding another verb



add

- ◆ Creates new entries in the cluster database
- ◆ Examples:
 - ➔ Hosts
 - ➔ Appliances
 - ➔ Rolls





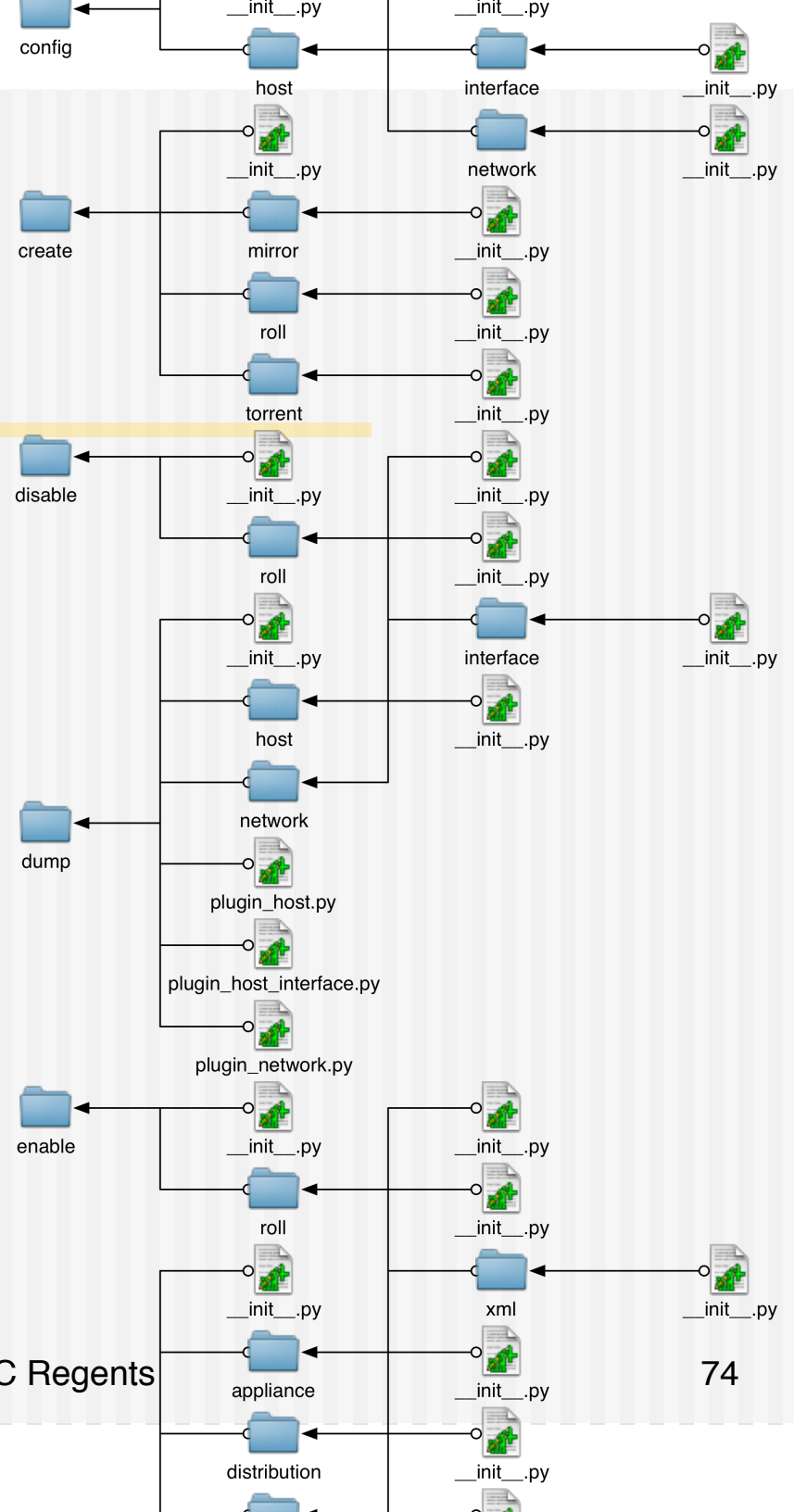
rocks add distribution

```
1: import rocks.commands
2:
3: class Command(rocks.commands.DistributionArgumentProcessor,
4:               rocks.commands.add.command):
5:     """
6:     Add a distribution specification to the database.
7:
8:     <arg type='string' name='distribution'>
9:     Name of the new distribution.
10:    </arg>
11:
12:    <example cmd='add distribution rocks-dist'>
13:    Adds the distribution named "rocks-dist" into the database.
14:    </example>
15:    """
16:
17:    def run(self, params, args):
18:
19:        if len(args) != 1:
20:            self.abort('must supply one distribution')
21:        dist = args[0]
22:
23:        if dist in self.getDistributionNames():
24:            self.abort('distribution "%s" exists' % dist)
25:
26:        self.db.execute("""insert into distributions (name) values
27:                        ('%s')""" % dist)
28:
29:
```



dump

- ◆ Returns cluster database information in the form of rocks command lines
- ◆ Examples:
 - ➔ Hosts
 - ➔ Network
- ◆ Same as `-dump` flag on `insert-ethers`





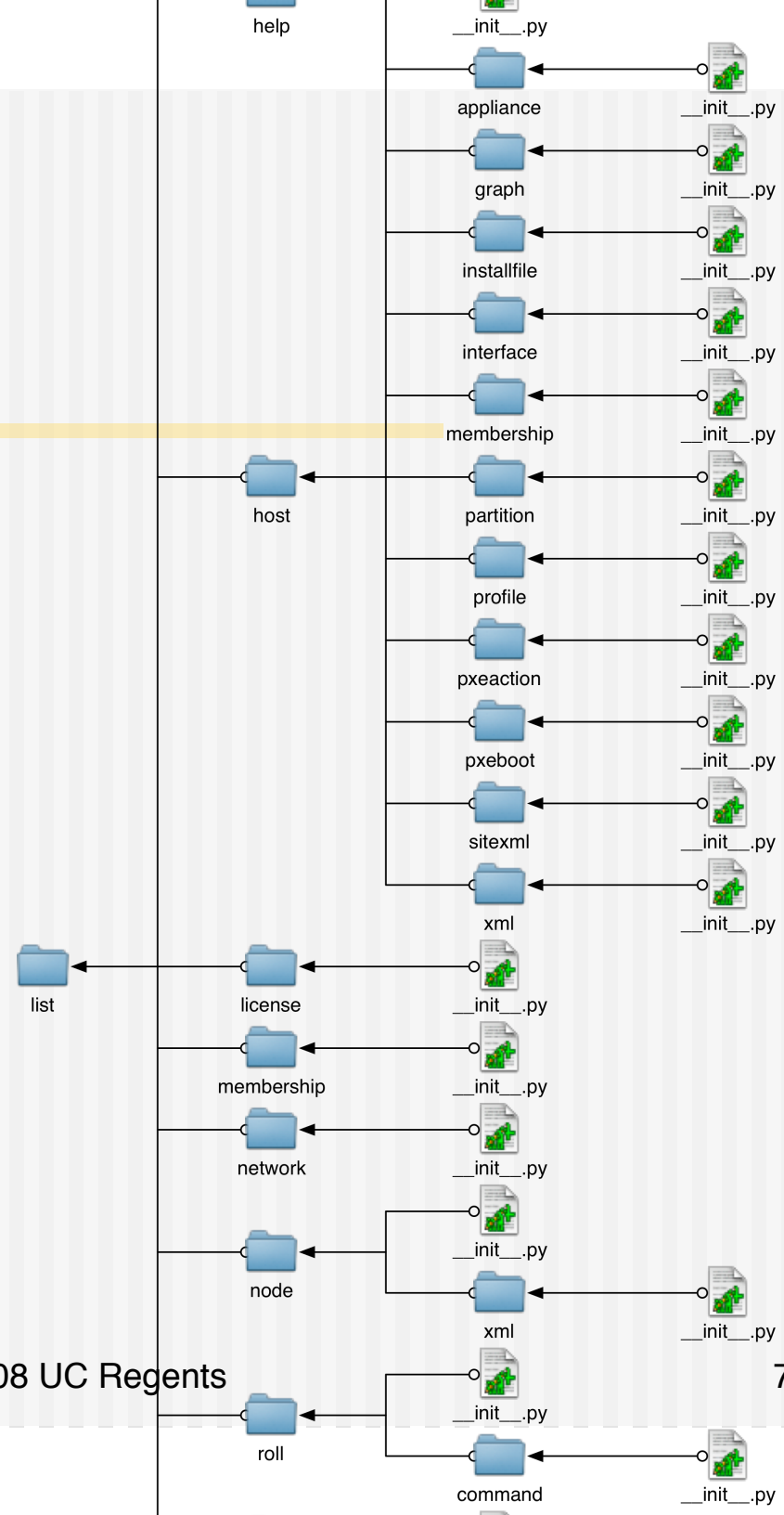
rocks dump host

```
# rocks dump host
/opt/rocks/bin/rocks add host vizagra cpus=1 rack=0 rank=0 membership="Frontend"
/opt/rocks/bin/rocks add host tile-0-1 cpus=2 rack=0 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-0-0 cpus=2 rack=0 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-0-2 cpus=2 rack=0 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-0-3 cpus=2 rack=0 rank=3 membership="Tile"
/opt/rocks/bin/rocks add host tile-1-3 cpus=2 rack=1 rank=3 membership="Tile"
/opt/rocks/bin/rocks add host tile-1-2 cpus=2 rack=1 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-1-1 cpus=2 rack=1 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-1-0 cpus=2 rack=1 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-2-0 cpus=2 rack=2 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-2-1 cpus=2 rack=2 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-2-2 cpus=2 rack=2 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-2-3 cpus=2 rack=2 rank=3 membership="Tile"
/opt/rocks/bin/rocks add host tile-3-0 cpus=2 rack=3 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-3-1 cpus=2 rack=3 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-3-2 cpus=2 rack=3 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-3-3 cpus=2 rack=3 rank=3 membership="Tile"
/opt/rocks/bin/rocks add host tile-4-0 cpus=2 rack=4 rank=0 membership="Tile"
/opt/rocks/bin/rocks add host tile-4-1 cpus=2 rack=4 rank=1 membership="Tile"
/opt/rocks/bin/rocks add host tile-4-2 cpus=2 rack=4 rank=2 membership="Tile"
/opt/rocks/bin/rocks add host tile-4-3 cpus=2 rack=4 rank=3 membership="Tile"
```



list

- ◆ Reports information in human readable format
- ◆ No side-effects on the database
- ◆ Examples:
 - ⇒ Hosts
 - ⇒ Appliances
 - ⇒ Rolls





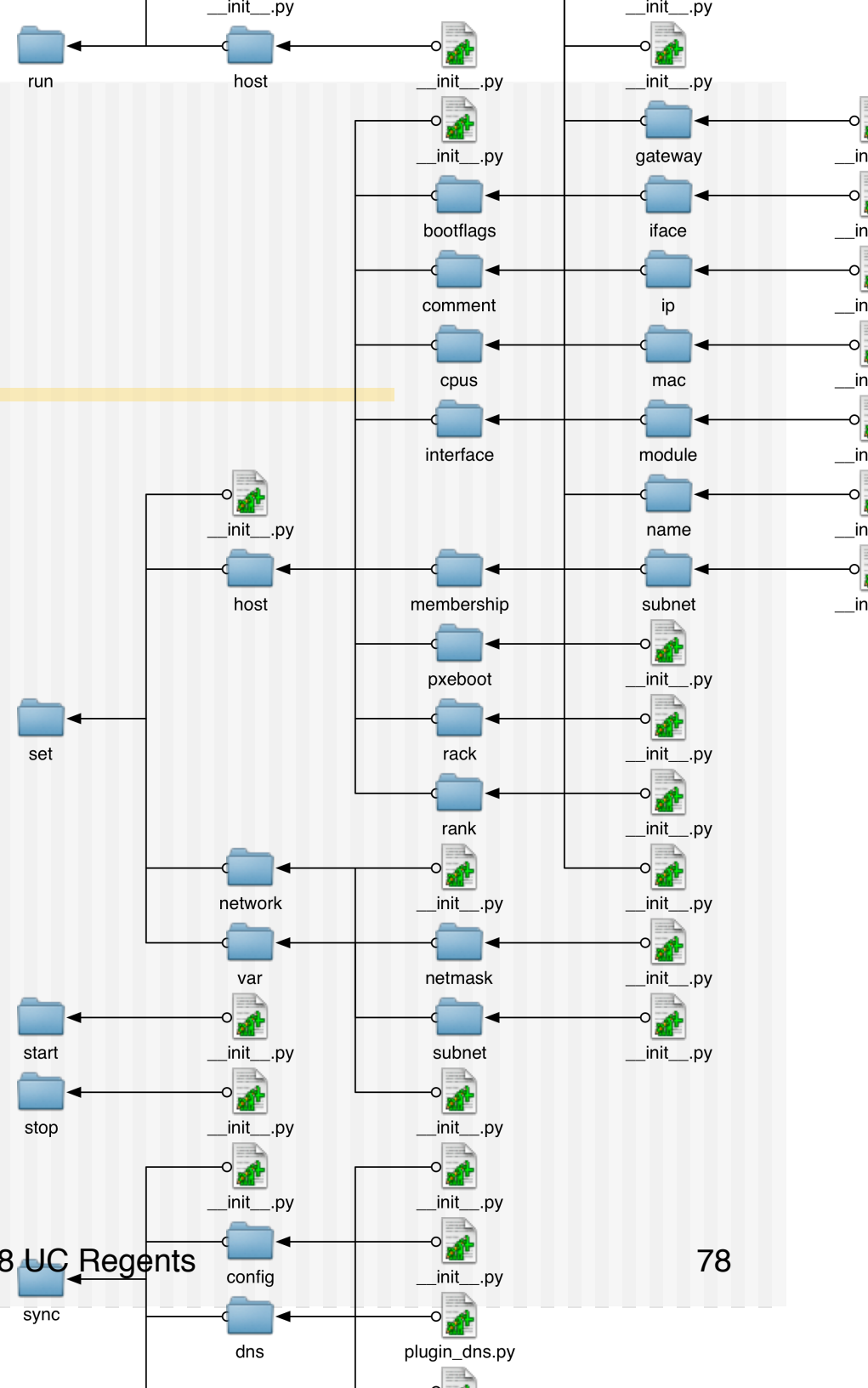
rocks list host

```
# rocks list host
HOST      MEMBERSHIP CPUS RACK RANK COMMENT
vizagra:  Frontend  1   0   0  -----
tile-0-1: Tile      2   0   1  -----
tile-0-0: Tile      2   0   0  -----
tile-0-2: Tile      2   0   2  -----
tile-0-3: Tile      2   0   3  -----
tile-1-3: Tile      2   1   3  -----
tile-1-2: Tile      2   1   2  -----
tile-1-1: Tile      2   1   1  -----
tile-1-0: Tile      2   1   0  -----
tile-2-0: Tile      2   2   0  -----
tile-2-1: Tile      2   2   1  -----
tile-2-2: Tile      2   2   2  -----
tile-2-3: Tile      2   2   3  -----
tile-3-0: Tile      2   3   0  -----
tile-3-1: Tile      2   3   1  -----
tile-3-2: Tile      2   3   2  -----
tile-3-3: Tile      2   3   3  -----
tile-4-0: Tile      2   4   0  -----
tile-4-1: Tile      2   4   1  -----
tile-4-2: Tile      2   4   2  -----
tile-4-3: Tile      2   4   3  -----
```



set

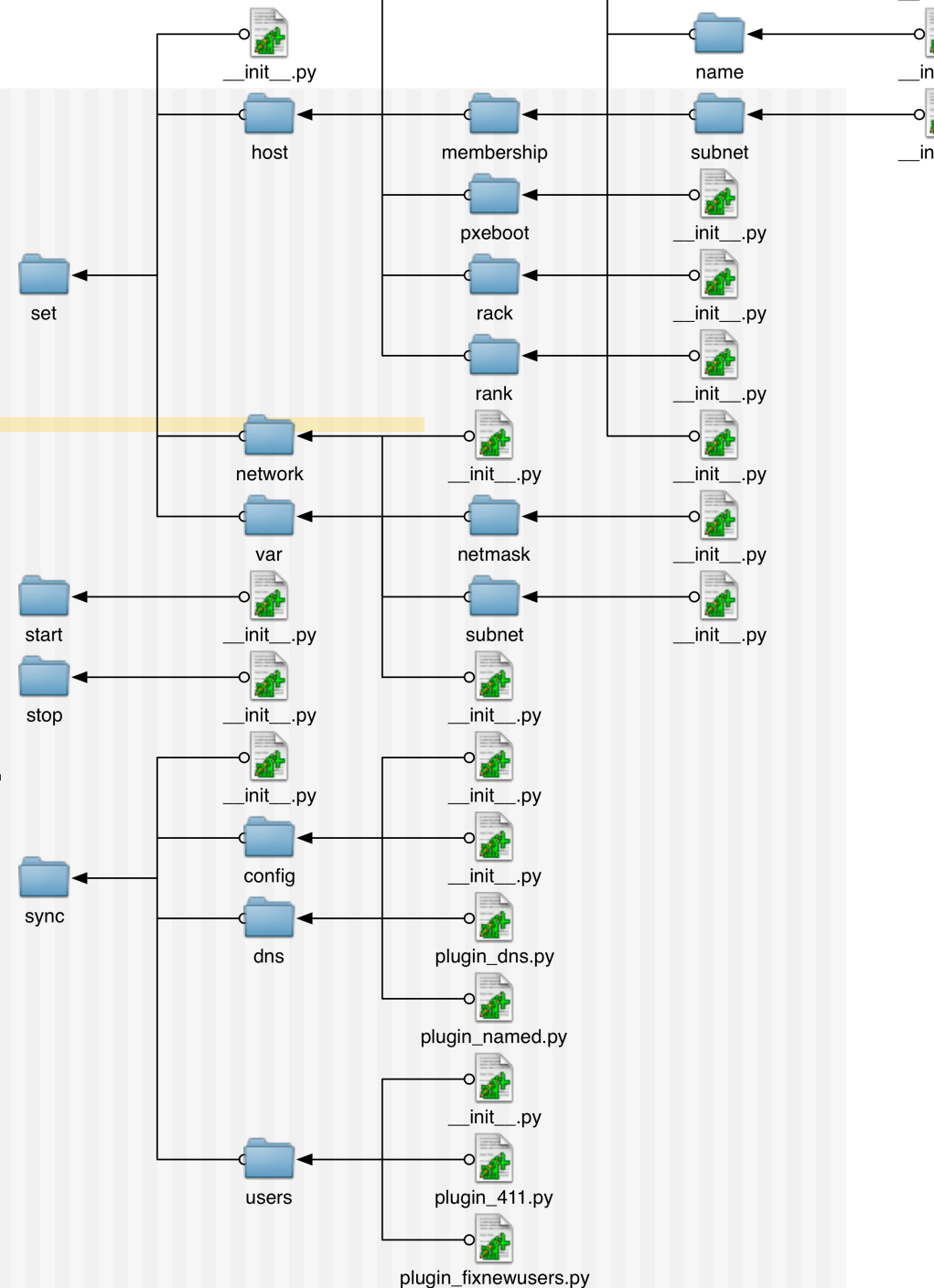
- ◆ Modifies entries in the cluster database
- ◆ Examples:
 - Network Interfaces
 - Appliance Assignment
 - Rack / Rank
- ◆ add-extra-nic
 - Rocks add host interface
 - Rocks set host interface





start / stop

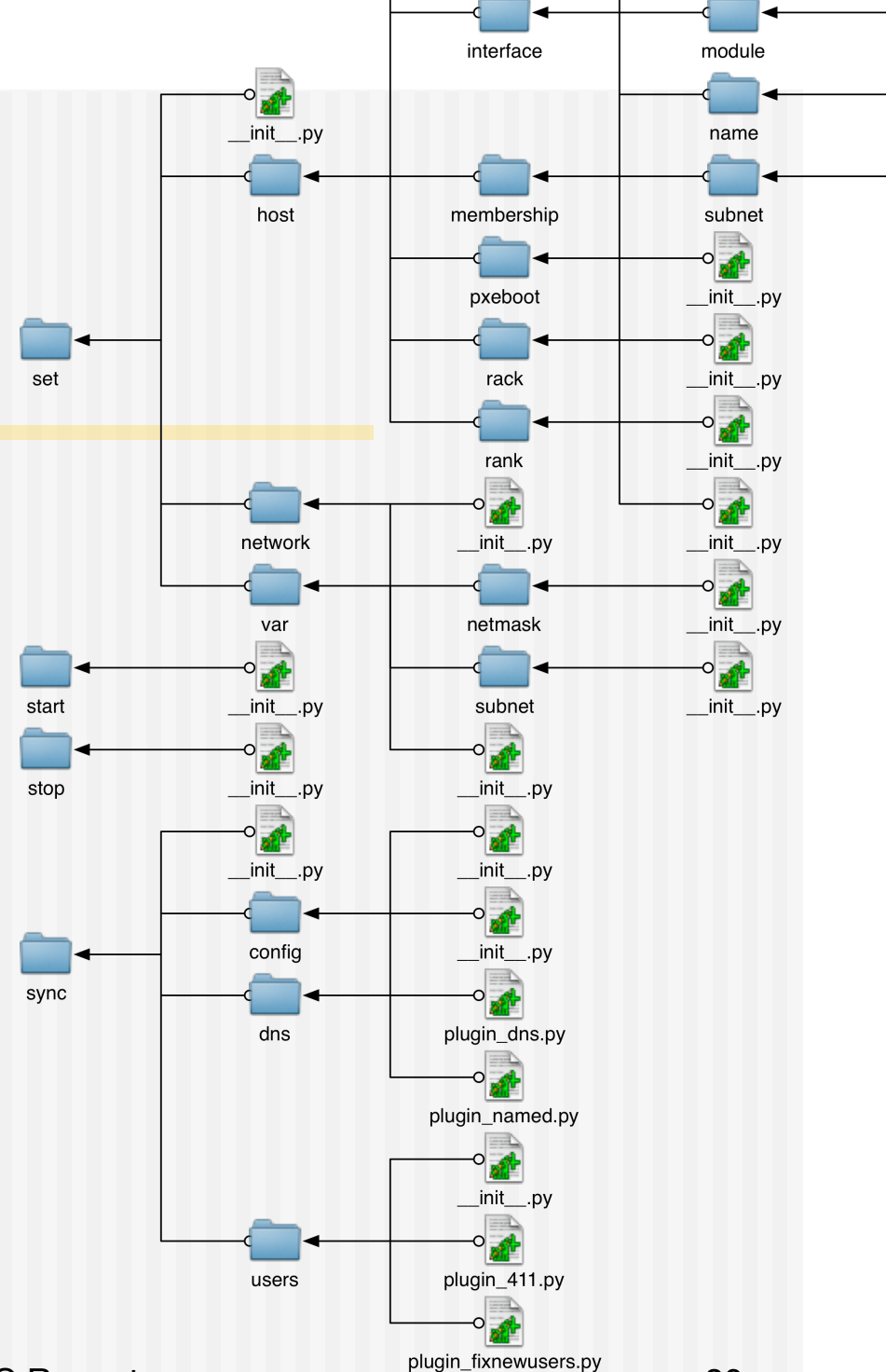
- ◆ Start and stop something
- ◆ NULL commands
- ◆ Reserve the verbs for use on other Rolls
- ◆ Think “abstract base class”





sync

- ◆ Synchronizes the database state to software configuration files
- ◆ Similar to the old “insert-ethers – update”





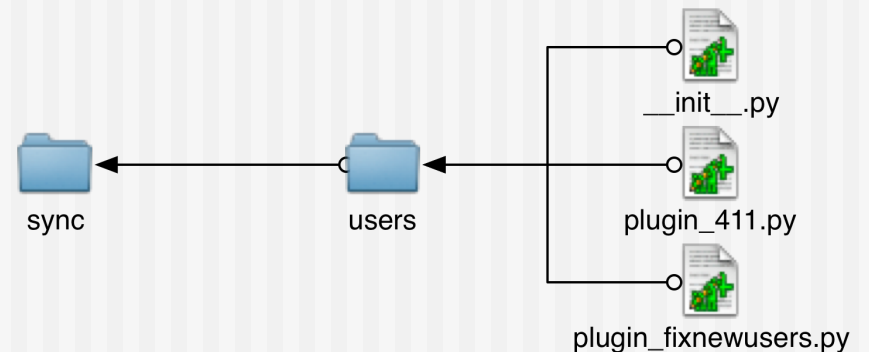
Extensibility

- ◆ New commands
 - ⇒ Add directories
 - ⇒ Add `__init__.py` code
- ◆ Existing commands
 - ⇒ Some commands can be extended
 - ⇒ Plugins



rocks sync users

- ◆ Run after useradd
 - Populate auto.home
 - Cleanup password file
 - Send 411 files
- ◆ Two plugins
 - Fixnewusers
 - 411
- ◆ Partial Ordering
- ◆ Other Rolls can add more plugins to this command
- ◆ Command must be design for plugins (not default)





__init__.py

```
1: import rocks.commands
2:
3:
4: class Command(rocks.commands.sync.command):
5:     """
6:     Update all user-related files (e.g., /etc/passwd, /etc/shadow, etc.)
7:     on all known hosts. Also, restart autofs on all known hosts.
8:
9:     <example cmd='sync users'>
10:    Send all user info to all known hosts.
11:    </example>
12:    """
13:
14:    def run(self, params, args):
15:        self.runPlugins()
16:
```



411 plugin

```
1: import os
2: import rocks.commands
3:
4: class Plugin(rocks.commands.Plugin):
5:     """Force a 411 update and re-load autofs on all nodes"""
6:
7:     def provides(self):
8:         return '411'
9:
10:    def requires(self):
11:        return ['fixnewusers']
12:
13:    def run(self, args):
14:        #
15:        # force the rebuild of all files under 411's control
16:        #
17:        for line in os.popen('make -C /var/411 force').readlines():
18:            self.owner.addText(line)
19:
20:        #
21:        # restart autofs on all known hosts
22:        #
23:        cmd = '/opt/rocks/bin/tentakel "service autofs reload"'
24:        for line in os.popen(cmd).readlines():
25:            self.owner.addText(line)
26:
27:
```



auto.home / passwd plugin

```
1: import os
2: import string
3: import rocks.commands
4:
5: class Plugin(rocks.commands.Plugin):
6:     """Relocates home directories to /export and fixes autofs.home"""
7:
8:     def provides(self):
9:         return 'fixnewusers'
10:
11:     def run(self, args):
12:         # scan the password file for any '/export/home' entries
13:         # this is the default entry as setup by useradd
14:         new_users = []
15:         default_dir = '/export/home/'
16:
17:         file = open('/etc/passwd', 'r')
18:
19:         for line in file.readlines():
20:             l = string.split(line[:-1], ':')
21:
22:             if len(l) < 6:
23:                 continue
24:
25:             username = l[0]
26:             homedir = l[5]
27:
28:             if homedir[:len(default_dir)] == default_dir:
29:                 new_users.append(username)
30:         file.close()
31:
32:         hostname = '%s.%s' % \
33:             (self.db.getGlobalVar('Kickstart', 'PrivateHostname'),
34:              self.db.getGlobalVar('Kickstart', 'PrivateDNSDomain'))
35:
36:         for user in new_users:
37:
38:             # for each new user, change their default directory to
39:             # /home/<username>
40:
41:             and = '/usr/sbin/usermod -d %s %s' % \
42:                 (os.path.join('/home', user), user)
43:             for line in os.popen(and).readlines():
44:                 self.owner.addText(line)
45:
46:             # then update the auto.home file
```



Argument Processing

- ◆ rocks <verb> <object...> <subject>
<params...>
- ◆ Subject is typed by first object
 - ⇒ host -> one or more hostname
 - ⇒ roll -> one or more roll names
- ◆ Params are in key=value form
- ◆ Same as -flag=value but easier to read



Helper classes and functions

- ◆ **ArgumentProcessors**
 - ⇒ Class to parse the subject in a standard way
 - ⇒ Exists for hosts, rolls, appliances, ...
- ◆ **Parameters Parsing**
 - ⇒ fillPositionalArgs
 - ⇒ fillParams



HostArgumentProcessor

- ◆ Command must derive from `rocks.commands.HostArgumentProcessor`
- ◆ `self.getHostnames(args)`
 - Return a list of hostname as they appear in the cluster database
 - If `args = None` all the host in the cluster are returned
 - `args` can also be a group
 - Rack0, rack1
 - Or an appliance type
 - Compute, Tile, ...

```

1: import rocks.commands
2:
3: class command(rocks.commands.HostArgumentProcessor,
4:               rocks.commands.list.command):
5:     pass
6:
7: class Command(command):
8:     """
9:     List the membership, CPU count, physical position info and comment for
10:    a list of hosts.
11:
12:    <arg optional='1' type='string' name='host' repeat='1'>
13:    Zero, one or more host names. If no host names are supplied, info about
14:    all the known hosts is listed.
15:    </arg>
16:
17:    <example cmd='list host compute-0-0'>
18:    List info for compute-0-0.
19:    </example>
20:
21:    <example cmd='list host'>
22:    List info for all known hosts.
23:    </example>
24:    """
25:
26:    def run(self, params, args):
27:        self.beginOutput()
28:
29:        for host in self.getHostnames(args):
30:            self.db.execute("""select m.name, n.cpus,
31:                               n.rack, n.rank, n.comment from
32:                               nodes n, memberships m where
33:                               n.membership=m.id and n.name='%s'""" % host)
34:            self.addOutput(host, self.db.fetchone())
35:
36:        self.endOutput(header=['host', 'membership',
37:                               'cpus', 'rack', 'rank', 'comment'])
38:

```



args = None

```
# rocks list host
```

HOST	MEMBERSHIP	CPUS	RACK	RANK	COMMENT
vizagra:	Frontend	1	0	0	-----
tile-0-1:	Tile	2	0	1	-----
tile-0-0:	Tile	2	0	0	-----
tile-0-2:	Tile	2	0	2	-----
tile-0-3:	Tile	2	0	3	-----
tile-1-3:	Tile	2	1	3	-----
tile-1-2:	Tile	2	1	2	-----
tile-1-1:	Tile	2	1	1	-----
tile-1-0:	Tile	2	1	0	-----
tile-2-0:	Tile	2	2	0	-----
tile-2-1:	Tile	2	2	1	-----
tile-2-2:	Tile	2	2	2	-----
tile-2-3:	Tile	2	2	3	-----



args = list of hosts

```
# rocks list host tile-0-0 10.255.255.253 tile-3-0.local
HOST      MEMBERSHIP CPUS RACK RANK COMMENT
tile-0-0: Tile      2    0    0    -----
tile-0-1: Tile      2    0    1    -----
tile-3-0: Tile      2    3    0    -----
```



args = rack

```
# rocks list host rack2
```

HOST	MEMBERSHIP	CPUS	RACK	RANK	COMMENT
tile-2-0:	Tile	2	2	0	-----
tile-2-1:	Tile	2	2	1	-----
tile-2-2:	Tile	2	2	2	-----
tile-2-3:	Tile	2	2	3	-----



args = appliance type

```
# rocks list host tile
```

HOST	MEMBERSHIP	CPUS	RACK	RANK	COMMENT
tile-0-0:	Tile	2	0	0	-----
tile-0-1:	Tile	2	0	1	-----
tile-0-2:	Tile	2	0	2	-----
tile-0-3:	Tile	2	0	3	-----
tile-1-0:	Tile	2	1	0	-----
tile-1-1:	Tile	2	1	1	-----
tile-1-2:	Tile	2	1	2	-----
tile-1-3:	Tile	2	1	3	-----
tile-2-0:	Tile	2	2	0	-----
tile-2-1:	Tile	2	2	1	-----



Any combination is fine

```
# rocks list host tile-2-0 rack1 frontend
HOST      MEMBERSHIP CPUS RACK RANK COMMENT
tile-1-0:  Tile      2   1   0  -----
tile-1-1:  Tile      2   1   1  -----
tile-1-2:  Tile      2   1   2  -----
tile-1-3:  Tile      2   1   3  -----
tile-2-0:  Tile      2   2   0  -----
vizagra:   Frontend   1   0   0  -----
```



ArgumentProcessors

Class Name	Helper Function
ApplianceArgumentProcessor	getApplianceNames
DistributionArgumentProcessor	getDistributionNames
HostArgumentProcessors	getHostnames
MembershipArgumentProcessor	getMembershipNames
NetworkArgumentProcessor	getNetworkNames
RollArgumentProcessor	getRollNames



RollArgumentProcessor

```
1: import os
2: import stat
3: import time
4: import sys
5: import string
6: import rocks.commands
7:
8:
9: class Command(rocks.commands.RollArgumentProcessor,
10:               rocks.commands.list.command):
11:     """
12:     List the status of available rolls.
13:
14:     <arg optional='1' type='string' name='roll' repeat='1'>
15:     List of rolls. This should be the roll base name (e.g., base, hpc,
16:     kernel). If no rolls are listed, then status for all the rolls are
17:     listed.
18:     </arg>
19:
20:     <example cmd='list roll kernel'>
21:     List the status of the kernel roll
22:     </example>
23:
24:     <example cmd='list roll'>
25:     List the status of all the available rolls
26:     </example>
27:     """
28:
29:     def run(self, params, args):
30:
31:         self.beginOutput()
32:         for (roll, version) in self.getRollNames(args, params):
33:             self.db.execute("""select version, arch, enabled from
34:                             rolls where name='%s' and version='%s'""" %
35:                             (roll, version))
36:             for row in self.db.fetchall():
37:                 self.addOutput(roll, row)
38:
39:         self.endOutput(header=['name', 'version', 'arch', 'enabled'],
40:                        trimOwner=0)
41:
```



No Parameter

```
# rocks list roll
```

NAME	VERSION	ARCH	ENABLED
viz:	5.0	i386	yes
sge:	5.0	i386	yes
kernel:	5.0	i386	yes
updates:	5.1	i386	yes
java:	4.3.2	i386	yes
xen:	5.0	i386	yes
CentOS:	5.1	i386	yes
ganglia:	5.0	i386	yes
web-server:	5.0	i386	yes
base:	5.0	i386	yes



Version Parameter

```
# rocks list roll version=4.3.2  
NAME    VERSION ARCH  ENABLED  
java: 4.3.2    i386 yes
```




Summary

- ◆ ArgumentProcessors standardize the handling of command line subjects
- ◆ Calling the helper function with an empty list returns all subject in the database
- ◆ HostArgumentProcessor knows about more than just host names
- ◆ RollArgumentProcessor can filter on versions



fillParams

- ◆ Create local variables based on command parameters (key=value)
- ◆ Argument a list of (key, default) tuples
- ◆ If the parameter is not found on the command line the default value is used

```
32: generates torrent files for every file in the RMS directory.  
33: </example>  
34: """
```

```
35:  
36: def maketorrent(self, filename, data):  
37:     info = {}  
38:     info['length'] = os.stat(filename)[stat.ST_SIZE]  
39:     info['name'] = os.path.basename(filename)
```

rocks create torrent

```
40:  
41:     data['info'] = info  
42:     encoded = BitTorrent.bencode(bencode(data))
```

```
43:  
44:  
45:     file = open('%s.torrent' % (filename), 'w')  
46:     file.write(encoded)  
47:     file.close()
```

```
48:  
49:  
50: def run(self, params, args):
```

```
51:  
52:     if len(args) != 1:  
53:         self.abort('must supply one file')  
54:     filename = args[0]
```

```
55:  
56:     (timestamp, ) = self.fillParams([('timestamp', time.time())])
```

```
57:     try:  
58:         creation_date = int(timestamp)  
59:     except:  
60:         creation_date = int(time.time())
```

```
61:  
62:     data = {}
```

```
63:  
64:     #
```

```
65:     # announce string
```

```
66:     #
```

```
67:     localhost = self.db.getGlobalVar('Kickstart', 'PrivateAddress')
```

```
68:     data['announce'] = 'http://%s:7625/announce' % (localhost)
```

```
69:  
70:     data['creation date'] = creation_date
```

```
71:  
72:     #
```

rocks add host

```
73:         basename, rack, rank = host.split('-')
74:         self.db.execute("""select m.name from
75:             appliances a, memberships m where
76:             a.name="%s" and m.appliance=a.id""" % basename)
77:         membership, = self.db.fetchone()
78:         rack = int(rack)
79:         rank = int(rank)
80:     except:
81:         membership = None
82:         rack = None
83:         rank = None
84:
85:     # fillParams with the above default values
86:
87:     (membership, numCPUs, rack, rank) = self.fillParams(
88:         [('membership', membership),
89:         ('cpus', 1),
90:         ('rack', rack),
91:         ('rank', rank)])
92:
93:     if not membership:
94:         self.abort('membership not specified')
95:     if rack == None:
96:         self.abort('rack not specified')
97:     if rank == None:
98:         self.abort('rank not specified')
99:
100:     self.db.execute("""insert into nodes
101:         (site, name, membership, cpus, rack, rank)
102:         values
103:         (0,
104:         '%s',
105:         (select id from memberships where name='%s'),
106:         '%d',
107:         '%d',
108:         '%d')""" %
109:         (host, membership, int(numCPUs), int(rack), int(rank)))
110:
111:
```



fillPositionalArgs

- ◆ Allows for parameters to have implied keys (just values on command line)
- ◆ This is an optimization for ease of use, not ease of software
- ◆ Argument is a list of keys
 - No default value processing, if a key is specified it is required
 - Use this only when a parameter is required
- ◆ Example:

```
# rocks set network netmask optiputer netmask=255.255.255.0  
# rocks set network netmask optiputer 255.255.0.0
```

rocks set network netmask

```
11: </arg>
12:
13: <arg type='string' name='netmask'>
14: Netmask that named networks should have.
15: </arg>
16:
17: <param type='string' name='netmask'>
18: Can be used in place of netmask argument.
19: </param>
20:
21: <example cmd='set network netmask optiputer 255.255.255.0'>
22: Sets the netmask for the "optiputer" network to a class-c address
23: space.
24: </example>
25:
26: <example cmd='set network netmask optiputer netmask=255.255.255.0'>
27: Same as above.
28: </example>
29:
30: <example cmd='set network netmask optiputer cavewave 255.255.0.0'>
31: Sets the netmask for the "optiputer" and "cavewave" networks to
32: a class-b address space.
33: </example>
34:
35: <related>add network</related>
36: <related>set network subnet</related>
37: """
38:
39: def run(self, params, args):
40:     (args, netmask) = self.fillPositionalArgs(('netmask',))
41:
42:     if not len(args):
43:         self.abort('must supply network')
44:     if not netmask:
45:         self.abort('must supply netmask')
46:
47:     for network in self.getNetworkNames(args):
48:         self.db.execute("""update subnets set netmask='%s' where
49:             subnets.name='%s'""" % (netmask, network))
50:
```

```

32: Sets the MAC Address for the eth1 device on host compute-0-0.
33: </example>
34:
35: <example cmd='set host interface mac compute-0-0 iface=eth1 mac=00:0e:0c:a7:5d:ff'>
36: Same as above.
37: </example>
38:
39: <example cmd='set host interface mac compute-0-0 iface=eth1 mac=NULL'>
40: clears the mac address from the database
41: </example>
42:
43: <!-- cross refs do not exist yet
44: <related>set host interface iface</related>
45: <related>set host interface ip</related>
46: <related>set host interface gateway</related>
47: <related>set host interface module</related>
48: -->
49: <related>add host</related>
50: """

```

rocks set host interface

```

51:
52: def run(self, params, args):
53:
54:     (args, iface, mac) = self.fillPositionalArgs(('iface', 'mac'))
55:
56:     hosts = self.getHostnames(args)
57:
58:     if len(hosts) != 1:
59:         self.abort('must supply one host')
60:     if not iface:
61:         self.abort('must supply iface')
62:     if not mac:
63:         self.abort('must supply mac')
64:
65:     for host in hosts:
66:         self.db.execute("""update networks, nodes set
67:             networks.mac=NULLIF('%s', 'NULL') where
68:             nodes.name='%s' and networks.node=nodes.id and
69:             (networks.device='%s' or networks.mac='%s')""" %
70:             (mac, host, iface, iface))
71:

```



Help and Docstrings

- ◆ The command line is the documentation
 - ⇒ No more out of date man pages
 - ⇒ Still needs a cookbook document, but reference is part of the code
- ◆ We've been looking at this all session
- ◆ Class docstring `"""text"""`
- ◆ Command line has an XML format



```
# rocks list roll help  
rocks list roll [roll]...
```

Description:

List the status of available rolls.

Arguments:

[roll]

List of rolls. This should be the roll base name (e.g., base, hpc, kernel). If no rolls are listed, then status for all the rolls are listed.


Examples:

```
$ rocks list roll kernel
```

List the status of the kernel roll

```
$ rocks list roll
```

List the status of all the available rolls



```

1: import os
2: import stat
3: import time
4: import sys
5: import string
6: import rocks.commands
7:
8:
9: class Command(rocks.commands.RollArgumentProcessor,
10:               rocks.commands.list.command):
11:     """
12:     List the status of available rolls.
13:
14:     <arg optional='1' type='string' name='roll' repeat='1'>
15:     List of rolls. This should be the roll base name (e.g., base, hpc,
16:     kernel). If no rolls are listed, then status for all the rolls are
17:     listed.
18:     </arg>
19:
20:     <example cmd='list roll kernel'>
21:     List the status of the kernel roll
22:     </example>
23:
24:     <example cmd='list roll'>
25:     List the status of all the available rolls
26:     </example>
27:     """
28:
29:     def run(self, params, args):
30:
31:         self.beginOutput()
32:         for (roll, version) in self.getRollNames(args, params):
33:             self.db.execute("""select version, arch, enabled from
34:                             rolls where name='%s' and version='%s'""" %
35:                             (roll, version))
36:             for row in self.db.fetchall():
37:                 self.addOutput(roll, row)
38:
39:         self.endOutput(header=['name', 'version', 'arch', 'enabled'],
40:                        trimOwner=0)
41:

```

rocks list roll help



<arg>

◆ Attributes

- ⇒ name (required)
- ⇒ optional (default = “0”)
- ⇒ type (default = “string”)
- ⇒ repeat (default = “0”)

◆ Example:

```
<arg type='string' name='network' repeat='1'>
```

One or more named networks that should have the defined netmask.

```
</arg>
```



<param>

◆ Attributes

- ⇒ name (required)
- ⇒ optional (default = “1”)
- ⇒ type (default = “string”)
- ⇒ repeat (default = “0”)

◆ Example:

```
<param type='string' name='iface'>
```

Can be used in place of the iface argument.

```
</param>
```



<example>

◆ Attributes

⇒ cmd(required)

◆ Example:

```
<example cmd='set host interface mac compute-0-0  
eth1 00:0e:0c:a7:5d:ff'>
```

Sets the MAC Address for the eth1 device on host compute-0-0.

```
</example>
```



<related>

◆ Example

```
<related>set host interface iface</related>  
<related>set host interface ip</related>  
<related>set host interface gateway</related>  
<related>set host interface module</related>
```



Help

- ◆ rocks <verb> <object...> <subject> help
 - ⇒ Loads the command module
 - ⇒ Parses the XML docstring
 - ⇒ Format and output help as 80 column text
- ◆ Debug syntax with `format=` parameter



help format=raw

```
# rocks list roll help format=raw
1:
2: List the status of available rolls.
3:
4: <arg optional='1' type='string' name='roll' repeat='1'>
5: List of rolls. This should be the roll base name (e.g., base, hpc,
6: kernel). If no rolls are listed, then status for all the rolls are
7: listed.
8: </arg>
9:
10:<example cmd='list roll kernel'>
11:List the status of the kernel roll
12:</example>
13:
14:<example cmd='list roll'>
15:List the status of all the available rolls
16:</example>
```




Help format=parsed

```
# rocks list roll help format=parsed
{'related': [], 'example': [(u'list roll kernel', u'\t\t\n\tList the
status of the kernel roll\n\t'), (u'list roll', u'\n\tList the status
of all the available rolls\n\t')], 'description': u'\n\tList the status
of available rolls.\n\t\n\t', 'param': [], 'arg': [(u'roll',
u'string', 1, 1), u'\n\tList of rolls. This should be the roll base
name (e.g., base, hpc,\n\tkernel). If no rolls are listed, then status
for all the rolls are\n\tlisted.\n\t')]]}
```



Docbook

◆ Roll Usersguide Command Reference is generated automatically

```
# rocks list roll help format=docbook
<section id="rocks-list-roll" xreflabel="list roll">
<title>list roll</title>
<cmdsynopsis>
  <command>rocks list roll</command>
  <arg rep="repeat" choice="opt">roll</arg>
</cmdsynopsis>
<para>
```

List the status of available rolls.

```
</para>
<variablelist><title>arguments</title>
<varlistentry>
```

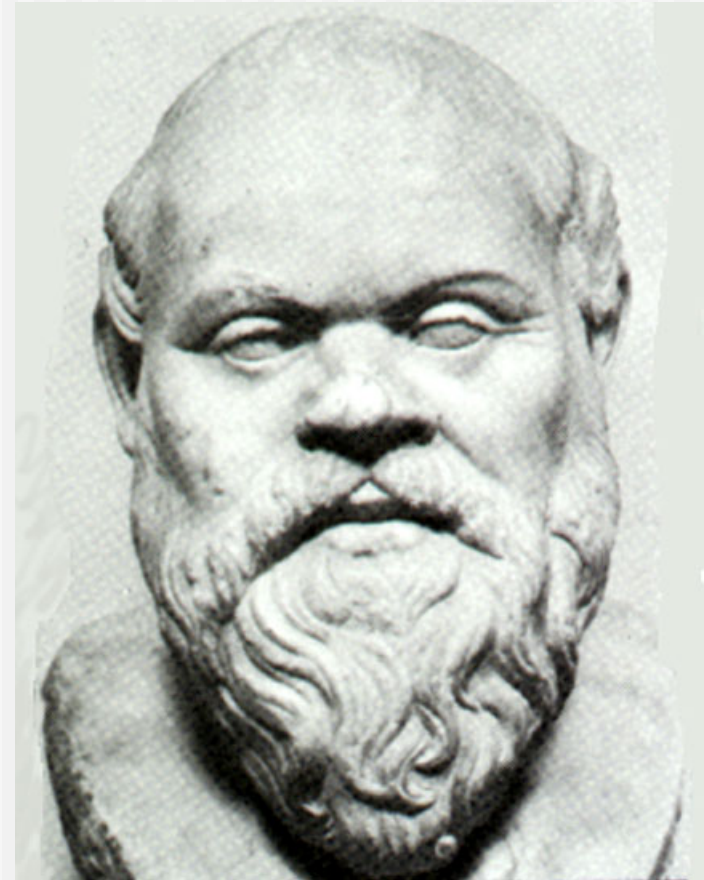


AVOIDING BECOMING A DEVELOPER



Philosophy

- ◆ All software is installed on the local disk
- ◆ Does not require NFS or non-scalable diskless technologies
- ◆ Use the native OS packager for everything
 - Linux = rpm
 - Solaris = pkg





Violate the Rules

- ◆ You just need a few packages added and cannot find or build packages
- ◆ You want this only on your cluster and not on several clusters
- ◆ You still want to avoid NFS and benefit from Rocks management





Get a Directory Tree

- ◆ Build your software from source and install on the frontend
 - ⇒ configure
 - ⇒ make
 - ⇒ install
- ◆ Or, just untar a binary bundle



CREATE PACKAGE

```
rocks create package  
  <path>  
  <package-name>
```

```
rocks create package  
  /opt/mx  
  mx
```



Done

```
# rpm -qip mx-1.0-1.x86_64.rpm
```

Name	: mx	Relocations:	(not relocatable)
Version	: 1.0	Vendor:	Rocks Clusters
Release	: 1	Build Date:	Tue 12 May 2009 04:40:00 PM
PDT			
Install Date:	(not installed)	Build Host:	vizagra.rocksclusters.org
Group	: System Environment/Base	Source RPM:	mx-1.0-1.src.rpm
Size	: 17588899	License:	University of California
Signature	: (none)		
Summary	: A collection of Python software tools.		
Description :			
The mx extensions for Python are a collection of Python software tools which enhance Python's usability in many areas.			



ADDING YOUR PACKAGE TO COMPUTE NODES



Step 1: Contribute the RPM

- ◆ Your distribution looks for packages from Rolls and in a contrib area
- ◆ Copy your RPMS into contrib

```
cp mx-1.0-1.x86_64.rpm  
  /export/rocks/install/contrib/5.2/  
  x86_64/RPMS
```



Step 2: Extend XML

```
cd /export/rocks/install/site-  
profiles/5.2/nodes/
```

```
cp skeleton.xml  
    extend-compute.xml
```

```
vi extend-compute.xml
```



Add Package Tag

original

```
<kickstart>

<description>
Skeleton XML Node
</description>

<changelog>
</changelog>

<!--
<package></package>
-->

<post>
</post>

</kickstart>
```

modified (with mx)

```
<kickstart>

<description>
Skeleton XML Node
</description>

<changelog>
</changelog>

<package>mx</package>

<post>
</post>

</kickstart>
```



Step 3: Rebuild Distribution

- ◆ RPM package is already contributed
- ◆ XML node file is already extended
- ◆ Now we need to rebuild the dist
- ◆ Must be done in `/export/rocks/install`



CREATE DISTRO

```
cd /export/rocks/install
```

```
rocks create distro
```



Step 4: Re-install

(repeated material 3 slides)

◆ PXE Boot

- Network Boot is first in BIOS boot order
- Set Rocks Boot action to install
- Reboot the host

◆ Otherwise use old rocks commands or just hard power cycle the host.



SET HOST BOOT

```
rocks set host boot  
    <host>  
    action=<boot-action>
```

```
rocks set host boot  
    compute-0-0  
    action=install
```




RUN HOST

```
rocks run host
```

```
<host>
```

```
<command>
```

```
rocks run host
```

```
compute-0-0
```

```
/sbin/init 6
```



Should I Build a Roll?

contrib & site-profiles

- ◆ Fast and Easy
- ◆ Admin Friendly
- ◆ Difficult to share
- ◆ Difficult to backup/restore
- ◆ Frontend is your development host

Roll

- ◆ Takes about 1 day
- ◆ Developer Friendly
- ◆ Easy to share (.iso)
- ◆ Easy to backup/restore
- ◆ Frontend is your development host



Break Time
